Univerza v Ljubljani
Fakulteta za računalništvo in informatiko

Matija Volontar

Optimizacija dodeljevanja nalog v okviru oskrbovanja objektov s pomočjo pametnih mobilnih naprav

DIPLOMSKO DELO
VISOKOŠOLSKI STROKOVNI ŠTUDIJSKI PROGRAM PRVE STOPNJE
RAČUNALNIŠTVO IN INFORMATIKA

doc. dr. Damjan Vavpotič
MENTOR

Ljubljana, 2018

Tematika naloge:

V prvem delu diplomske naloge preučite in predstavite procese in sisteme, ki se uporabljajo v okviru oskrbovanja objektov, pri čemer se še posebno posvetite procesom za dodeljevanje nalog vzdrževalcem. Analizirajte možnosti za optimizacijo procesa dodeljevanja nalog z uvedbo ustrezne informacijske rešitve, ki bo vključevala uporabo pametnih mobilnih naprav ter identificirajte ključne funkcionalnosti takšne rešitve. Na podlagi analize preučite primernost morebitnih obstoječih namenskih informacijskih rešitev. V drugem delu naloge izdelajte delujoč prototip rešitve za izbrano podjetje in ga v njem preizkusite. Primerjajte učinkovitost izvajanja procesa dodeljevanje nalog v okviru oskrbovanja objektov na obstoječ način in na optimiziran način z uporabo pripravljenega prototipa. Rezultate dela kritično ovrednotite.

g

h

IZJAVA O AVTORSTVU DIPLOMSKEGA DELA

Spodaj podpisani Matija Volontar, vpisna številka 63140284, avtor pisnega zaključnega dela študija z naslovom:

Optimizacija dodeljevanja nalog v okviru oskrbovanja objektov s pomočjo pametnih mobilnih naprav (*angl. Optimization of task assignement in facility maintenance by using mobile technologies*)

IZJAVLJAM

1. da sem pisno zaključno delo študija izdelal samostojno pod mentorstvom doc. dr. Damjana Vavpotiča;
2. da je tiskana oblika pisnega zaključnega dela študija istovetna elektronski obliki pisnega zaključnega dela študija;
3. da sem pridobil vsa potrebna dovoljenja za uporabo podatkov in avtorskih del v pisnem zaključnem delu študija in jih v pisnem zaključnem delu študija jasno označil;
4. da sem pri pripravi pisnega zaključnega dela študija ravnal v skladu z etičnimi načeli in, kjer je to potrebno, za raziskavo pridobil soglasje etične komisije;
5. soglašam, da se elektronska oblika pisnega zaključnega dela študija uporabi za preverjanje podobnosti vsebine z drugimi deli s programsko opremo za preverjanje podobnosti vsebine, ki je povezana s študijskim informacijskim sistemom članice;
6. da na UL neodplačno, neizključno, prostorsko in časovno neomejeno prenašam pravico shranitve avtorskega dela v elektronski obliki, pravico reproduciranja ter pravico dajanja pisnega zaključnega dela študija na voljo javnosti na svetovnem spletu preko Repozitorija UL;
7. dovoljujem objavo svojih osebnih podatkov, ki so navedeni v pisnem zaključnem delu študija in tej izjavi, skupaj z objavo pisnega zaključnega dela študija.

V Ljubljani, dne 1. februarja 2018

Podpis študenta:

Univerza v Ljubljani
Fakulteta za računalništvo in informatiko

Matija Volontar

Optimizacija dodeljevanja nalog v okviru oskrbovanja objektov s pomočjo pametnih mobilnih naprav

POVZETEK

V začetku diplomskega dela predstavimo delovanje sistema SCADA in njegovo vlogo v okviru oskrbovanja objektov. Izpostavimo problem optimalnega dodeljevanja nalog osebju na terenu ob alarmu na objektu z uporabo omenjenega sistema. V nadaljevanju, na podlagi raziskave trga in sodelovanja z zaposlenimi v testnem podjetju, predstavimo seznam funkcionalnosti, brez katerih rešitev ne more učinkovito reševati omenjenega problema. Glede na število vsebovanih funkcionalnosti in glede na ustreznost uporabe rešitve v dejanskem podjetju, so izbrane in predstavljene rešitve, ki ta problem naslavljajo. Naslednje poglavje opisuje orodja, naprave in način razvoja prototipa, ki vsebuje spletno in mobilno aplikacijo, ter storitve v oblaku. Sledi primerjava učinkovitosti izvajanja poslovnega procesa na obstoječ način in na optimiziran način z uporabo razvitega prototipa ter sklep.

Ključne besede: sistemi SCADA, proces alarmiranja, optimalno obveščanje vzdrževalcev

University of Ljubljana
Faculty of Computer and Information Science

Matija Volontar

Optimization of task assignement in facility maintenance by using mobile technologies

ABSTRACT

In the beggining of thesis we present, how SCADA system works and its role in the facility maintenance. We point out the problem of optimal task assignment among personnel on field, in case of alarm at certain facility, using the above mentioned system. Further in the thesis, based on market research and cooperation with employees of the company, we present a list of functionalities without which solutions can not effectively solve the problem. Based on the number of functionalities contained in each solution and based on the adequacy of the solution for use in an actual company, the list of three solutions that adress the given problem, is presented. Next, thesis describes the tools, devices, and development techniques used to produce a protoype that includes mobile application, web application and cloud services. Lastly we present a comparison between business process efficiency using standard methods and developed prototype. At the end, conclusion is presented.

Key words: SCADA systems, alarming process, optimal informing of maintenance workers

ZAHVALA

Rad bi se zahvalil svojemu mentorju za strokovno svetovanje in potrpežljivost pri nastajanju diplomskega dela. Zahvala gre tudi staršem, ki so mi omogočili študij, in punci Zali za pomoč pri uporabi pravilne slovnice.

— Matija Volontar, Ljubljana, februar 2018.

KAZALO

Povzetek i

Abstract iii

Zahvala v

1 Uvod 3

2 Sistem SCADA 5

2.1 Arhitektura sistema SCADA 6

2.1.1 Merilne naprave 6

2.1.2 Komunikacijska omrežja 6

2.1.3 Podatkovna baza 6

2.2 Prednosti in slabosti sistema SCADA 7

2.3 Oskrbovanje objektov s pomočjo sistema SCADA 8

2.3.1 Upravljanje alarmov 8

2.3.2 Proces alarmiranja 9

3 Pregled sorodnih rešitev 11

3.1 Tasker 12

3.2 Mobilplan 13

3.3 Recursion 15

4 Rešitev za obveščanje vzdrževalcev 17

4.1 Pregled dela 18

4.2 Platforma Google Firebase 19

4.2.1 Hranjenje podatkov 20

4.2.2 Funkciji v oblaku . . . 22
4.3 Spletna aplikacija . . . 24
4.3.1 Spreminjanje stanja objekta . . . 24
4.3.2 Obvestilo o sprejemu zahtevka . . . 26
4.4 Mobilna aplikacija . . . 26
4.4.1 Povezava na Google Firebase . . . 28
4.4.2 Obvladovanje zahtevkov za delo . . . 29
4.4.3 Pošiljanje lokacije . . . 30

5 Preizkus rešitve v podjetju 33
5.1 Testno podjetje . . . 33
5.1.1 Komunikacije med objektom in sistemom SCADA . . . 34
5.1.2 Servisna služba . . . 35
5.1.3 Proces klasičnega obveščanja vzdrževalcev . . . 35
5.2 Sprememba poslovnega procesa . . . 37
5.3 Predstavitev in uvedba rešitve v podjetju . . . 38
5.4 Rezultati testiranja . . . 38

6 Sklepne ugotovitve 47

Seznam uporabljenih kratic

Kratica	**Angleško**	**Slovensko**
SCADA	supervisory control and data acquisition	sistem, namenjen nadzorovanju in krmiljenju tehnološkega procesa z računalnikom
PLC	programmable logic controller	programabilen logični krmilnik
I/O	input/output	vhodno/izhodna naprava
VMS	virtual memory system	navidezni pomnilniški sistem
LAN	local area network	lokalno omrežje
OPC	open platform communications	odprta komunikacijska platforma
RTDB	real time database	podatkovna baza, delujoča v realnem času
OPC AE	Open Platform Communications Alarm Events	odprtokodna komunikacijska platforma, namenjena delu z alarmi in dogodki
SQL	structured query language	strukturiran poizvedovalni jezik
RTU	remote terminal unit	oddaljen terminal
OS	operating system	operacijski sistem
JSON	JavaScript object notation	zapis javascript objektov
HTML	hypertext markup language	hipertekstni označevalni jezik
GPS	global positioning system	sistem globalnega določanja lokacije
GSON	Google script object notation	googlov zapis skriptnih objektov
GSM	global systems for mobile communications	globalni sistem za mobilno komunikacijo

1 Uvod

Mobilna tehnologija je v zadnjih letih doživela velik tehnološki napredek. Klasične mobilne telefone so zamenjali pametni mobilni telefoni. Pametni mobilni telefoni imajo za razliko od klasičnih veliko dodatnih funkcionalnosti, ki nam olajšujejo vsakdan. Mednje sodijo: pospeškometer, kamera, kompas, avtentikacija s prstnim odtisom, brezžično polnenje in druge. Prav tako pametni mobilni telefoni lahko sprejemajo signal GPS oz. opravljajo funkcijo iskanja lokacije. Število uporabnikov pametnih mobilnih telefonov se veča iz dneva v dan. Dejstvo potrjuje statistika, ki govori, da je do konca leta 2017 število uporabnikov pametnih telefonov naraslo na skoraj 2,4 milijard, kar predstavlja več kot tretjino svetovne populacije. [1]

Zanesljivost pametnih mobilnih naprav in namensko razvitih aplikacij je z razvojem zrasla do te mere, da se le to uporablja pri bolj zahtevnih, resnih in pomembnih opravilih, kot npr. pohitritev poslovnega procesa v podjetju. Različna podjetja vidijo mobilno tehnologijo kot orodje, ki ne le okrepi komunikacijo pač pa tudi olajšuje in pohitri poslovni proces. Mobilna tehnologija najbolj pride do izraza v podjetjih, ki se ukvarjajo s

prodajo različnih stvari, npr. nepremičnin, in v podjetjih, v katerih je glavna dejavnost oglaševanje. Kar 15% vseh aplikacij na trgu, ki se uporabljajo za podporo poslovnega procesa v podjetju, je namenjenih učinkovitemu delovanju zaposlenih na terenu. Sem spada tudi rešitev, ki je predstavljena v tej nalogi. Celostne rešitve na tem področju omogočajo funkcionalnosti kot npr.: učinkovito komunikacijo, ogled in razreševanje alarmov na terenu, ocenjevanje porabe potrošnega materiala, možnost zaznavanja napak v poslovnem procesu, spremljanje zaposlenih. [2]

Motiv dipomske naloge je bil izdelati celovito rešitev za pohitritev poslovnega procesa v podjetju Petrol d.d. Poslovni proces vsebuje prenos informacij med kontaktno osebo v pisarni in vzdrževalci na terenu. Komunikacija se odvija časovno potratno in tudi neučinkovito. Pred razvojem celovite rešitve, ki vsebuje mobilno in spletno aplikacijo ter logiko v oblaku, je bilo potrebno zelo dobro razumevanje celotnega poslovnega procesa. Podrobno smo pregledali, kaj na trgu že obstaja, in ugotovili, da je ponudba na spletu precej nasičena z aplikacijami, ki naslavljajo ta problem. Celovite rešitve, ki smo jih pregledali, se osredotočajo na bolj obširno sliko poslovnega procesa, zato zastavljeni problem, za razliko od naše rešitve, naslavljajo površno. Na podlagi znanja, ki smo ga pridobili ob sodelovanju s podjetjem in pri raziskavi trga, smo oblikovali množico funkcionalnosti, ki so predstavljale osnovo za razvoj celovite rešitve. Celovito rešitev smo po testiranju primerjali s prej uporabljenimi tehnikami reševanja problema komunikacije med kontaktno osebo in vzdrževalci na terenu, ki ne vključujejo uporabe mobilne aplikacije. Prav tako smo rešitev ocenili z izpraševanjem sodelujočih.

2 Sistem SCADA

Sistem SCADA pomeni (*angl. Supervisory Control and Data Acquisition*). Kot narekuje že samo ime, ni samo kontrolni sistem, ampak se osredotoča tudi na nadzorovanje. V računalništvu sistem SCADA umeščamo med porazdeljene arhitekture. Vsebuje naprave na terenu, ki komunicirajo z nadzornim računalnikom preko komunikacijske infrastrukture. [29]

Sistemi SCADA niso uporabljeni samo v industrijskih procesih, kot npr.: pri izdelavi in obdelavi jekla, proizvodnji in distribuciji elektrike, pri kemijskih procesih, pač pa tudi pri oskrbovanju objektov z vodo in toploto. Sistem SCADA se v zadnjem času vedno več pojavlja v avtomatiziranih procesih. [27] Število V/I naprav (*angl. Input/Output*) je v prej omenjenih primerih lahko od nekaj sto do nekaj deset tisoč. SCADA sistem se z razvojem informacijske tehnologije prilagaja hitro. Na trgu so tako rešitve, ki podpirajo več sto tisoč V/I naprav. [28]

2.1 Arhitektura sistema SCADA

V nadaljevanju so opisani bistveni deli sistema SCADA, ki so nujni za razumevanje procesa, na katerega se v tej nalogi osredotočamo bolj podrobno. Primer delovanja celotnega sistema v praksi, je prikazan na sliki 2.1.

2.1.1 Merilne naprave

Merilne naprave so krmilniki, namenjeni nadzorovanju določenih parametrov naprav. Največkrat uporabljene naprave so krmilniki PLC (*angl. Programable Logic Controler*) ali druge splošno uporabljene naprave, kot npr. krmilnik RTU (*angl. Remote Terminal Unit*). Krmilniki so povezani z napravo, ki jo nadzorujejo. Njihova glavna naloga je spremljanje nadzorovane naprave, zaznavanje sprememb in pošiljanje podatkov nadzornemu računalniku. [29, 31]

2.1.2 Komunikacijska omrežja

Krmilniki, razpršeni po terenu, so povezani z nadzornim računalnikom na različne načine, kot npr. z radijsko povezavo ali naročniško povezavo, ki je lahko podatkovna ali optična. Tako strojna kot tudi programska oprema krmilnikov in nadzornega računalnika mora zagotavljati, da način povezave ne vpliva na kakovost komunikacije. Konfiguracija komunikacijskega sistema je odvisna od:

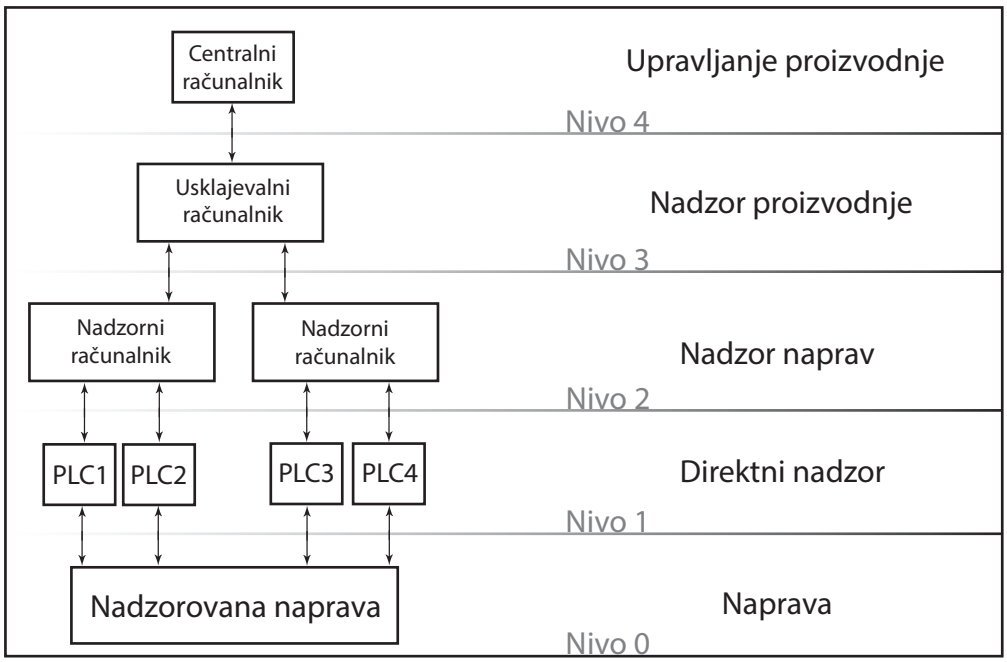
- števila krmilnikov in njihove lokacije
- zahtevane frekvence osveževanja podatkov
- razpoložljivih naprav in načinov za vzpostavitev komunikacije [31]

2.1.3 Podatkovna baza

Konfiguracijski podatki so za vmesnike, krmilnike in druge dele sistema SCADA shranjeni v podatkovni bazi. Podatkovna baza je fizično porazdeljena, vendar logično centralizirana, kar pomeni, da je uporabnikom porazdeljenost transparentna. Zaradi učinkovitega delovanja je celotna RTDB(*angl. Real Time Database*) shranjena še na strežniku. [31]

2.2 Prednosti in slabosti sistema SCADA

Velika prednost sistemov SCADA je avtomatizacija poslovnega procesa. Hkrati je želja po avtomatizaciji tudi razlog za razvoj in obstoj sistema. Dovoljuje nam, da se akcije samodejno sprožijo glede na različne dogodke. Skriptni jezik, ki ga nudi sistem SCADA, omogoča, da določimo, kateri dogodki sprožijo določene akcije. Primeri akcij so: pošiljanje e-pošte, zagon aplikacije, zapisovanje podatka v podatkovno bazo. [31] Program, ki skrbi za avtomatizacijo sistema, ne sme vsebovati napak, saj to lahko vodi v katastrofalne posledice. [30]



Slika 2.1 Primer delovanja sistema SCADA v proizvodnji [3]

V primerjavi z relejnimi sistemi, ki so se uporabljali nekaj desetletji nazaj, je sistem SCADA občutno cenejši za upravljanje in nadzorovanje. Omogoča enostavno programiranje in reprogramiranje glede na naše zahteve. Uporabljen je lahko tudi v bolj delikatnih situacijah, saj ob delovanju ne proizvaja toplote, tako kot se to dogaja pri relejnih sistemih. Poleg tega pa sistem SCADA porabi manj energije, zasede manj prostora in lahko izvaja aritmetične operacije. Ena večjih prednosti je ta, da je vsako spremembo moč opazovati na grafičnem vmesniku.

Slabosti v primerjavi s prej uporabljenimi relejnimi sistemi ni. Šibke točke SCADA sistema izhajajo iz posledic razvoja informacijske tehnologije. Sistem je lahko ranljiv za napade v vsaki točki poslovnega procesa. Največjo pozornost moramo usmeriti v točko, kjer je največja povezljivost in najmanjši nadzor dostopa. [4, 5] Avtomatizacija sistema SCADA se usmerja v zaznavanje nepravilnosti na napravah ter v obveščanje pristojnih o dogajanju, kar prikazuje diagram na sliki 2.2.

2.3 Oskrbovanje objektov s pomočjo sistema SCADA

V diplomskem delu se ukvarjamo s pohitritvijo procesa oskrbovanja objektov. Proces ima v podjetju Petrol d.d. po mnenju zaposlenih možnosti za izboljšave. S sodelovanjem z zaposlenimi v podjejtu smo ugotovili, kje v procesu so možnosti za izboljšave, in jih z našo rešitvijo naslovili. Procesu oskrbovanja objektov in na splošno sistemu SCADA se zato v tej nalogi posvečamo bolj podrobno.

Oskrbovani objekt je lahko stavba ali del stavbe, kjer s pomočjo sistema SCADA nadziramo oskrbovanje z vodo, toploto ali plinom. Pomembnejše funkcije sistema SCADA v tem sklopu so: pridobivanje podatkov iz oddaljenih naprav, prikaz podatkov uporabnikom sistema v razumljivi obliki, obvladovanje alarmov in krmiljenje. Naprav v objektu je lahko več. Primeri naprave so: obtočna črpalka, mešalni ventil, ogrevalni vir (toplotna črpalka, kotel, kogeneracijska naprava), naprave za preverjanje tlaka, naprave znotraj vodovodnih, odpadnovodnih in plinskih sistemov. Preko sistema SCADA pridobimo možnost popolnega upravljanja vsake izmed teh naprav na daljavo, grafičnega prikaza delovanja na sami lokaciji in na oddaljenem računalniku, možnost upravljanja z alarmi in hranjenje podatkov za kasnejše analize. [6]

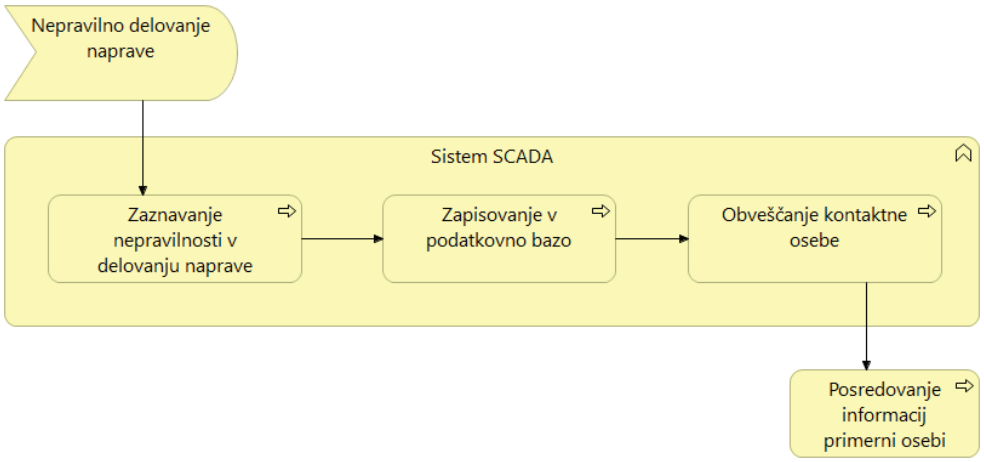
2.3.1 Upravljanje alarmov

Upravljanje alarmov temelji na preverjanju statusa povezanih krmilnikov. Upravljanje z alarmi se dogaja na strežniku. Bolj zapletene metode upravljanja z alarmi je možno razviti z ustvarjanjem izpeljanih parametrov. Obvladovanje alarmov se odvija centralno. To pomeni, da podatki obstajajo samo na enem mestu in vsi uporabniki spremljajo isti podatek. Alarmi se glede na prioriteto razlikujejo. Na splošno so razdeljeni v tri nivoje. Programska oprema ponuja možnost grupiranja alarmov. Tako lahko upravljamo s posameznim alarmom ali pa s skupino alarmov z isto prioriteto. [31] Sproženi alarm

ne pomeni, da ima naprava napako. Sistem lahko nastavimo tako, da se alarm sproži z določeno frekvenco ali ob določenemu dogodku. Na ta način lahko res natančno spremljamo delovanje naprav in ukrepamo pravočasno. [30]

2.3.2 Proces alarmiranja

Proces alarmiranja igra pomembno vlogo v sistemih SCADA. Sistem konstantno preverja, če je določena vrednost izven normalnih meja. V kolikor se to zgodi, sistem SCADA sporoči napako v centralni računalnik, ki je normalno stacioniran v podjetju. Podatek začne svojo pot na krmilniku, ki nadzira napravo, in nato preko določenega omrežja k odjemalcu. Preden podatek doseže odjemalca, se v podatkovno bazo zapiše časovni žig, narava alarma in ime naprave, ki ga je povzročila. Indikatorji alarma na samem računalniku so lahko naslednji: zvočni signal, opozorilno okno na ekranu, spremenjena barva naprave na grafičnemu prikazu sistema, e-pošta, sms sporočilo. Namen indikatorja je pritegniti pozornost kontaktne osebe, ki bo lahko informacijo o nastalem alarmu posredovala naprej. [7]



Slika 2.2 Primer splošnega procesa alarmiranja sistemov SCADA

Informacije o alarmu se lahko posreduje določeni skupini vzdrževalev ali posamezni osebi, ki alarm razume in ga zna odpraviti. Informacije so posredovane preko sms sporočila, e-pošte, klica ali pozivnika. Tukaj pa se pojavi problem v procesu alarmiranja, ki ga v tej nalogi tudi rešujemo. Člani skupine prejemnikov alarma se morajo med sabo sporazumeti in določiti osebo, ki bo alarm poizkušala odpraviti. To pa brez ustrezne informacijske podpore terja precej časa. Vsi prejemniki so lahko zasedeni s prej sproženimi alarmi in

se na poslano obvestilo ne odzovejo. Kontakta oseba nima informacije, kdo je prejel obvestilo o alarmu, niti o tem, kdo namerava alarm rešiti, razen seveda v primeru klica. S tem v procesu alarmiranja nastanejo velike izgube časa in kaos. V trenutku je lahko več alarmov, ki jih je treba nemudoma nasloviti. Kontaktna oseba nima podatka o tem, kje se nahajajo vzdrževalci, zato obvestilo o alarmu lahko pošlje vzdrževalcu, ki se nahaja precej stran, hkrati pa obstaja prost vzdrževalec, ki je v bližini objekta z napako. [8]

3 Pregled sorodnih rešitev

Celostne rešiteve, ki se osredotočajo predvsem na problem komunikacije med kontaktno osebo v pisarni in vzdrževalci na terenu, so redke. Rešitve zaobjamejo precej večji problem, kot je npr. celotno upravljanje z vzdrževalci. To med drugim vključuje tudi beleženje delovnih ur, načrtovanje odmorov, ugotavljanje vzorca napak in pripravljanje obračuna delovnih časov. Ravno zaradi obsežnosti podobnih celostnih rešitev se problem, ki ga rešujemo mi, rešuje pomanjkljivo.

Iskanja obstoječih rešitev smo se lotili glede na funkcionalnosti, ki jih aplikacija za učinkovito reševanje omenjenega problema mora vsebovati. Ključne funkcionalnosti smo izbrali glede na pogovor z zaposlenimi in glede na obstoječe rešitve po spletu. Spodaj je prikazana unija omenjenih izborov funkcionalnosti. Če se poleg funkcionalnosti v oklepaju nahaja simbol (*), pomeni, da je bila funkcionalnost izbrana na podlagi obstoječih rešitev. Funkcionalnosti brez oznake pa so bile izbrane na podlagi zahtev v podjejtu.

Ključne funkcionalnosti rešitve kot celote so naslednje:

1. Možnost prejemanja zahtevkov za delo
2. Možnost sprejemanja in zavračanja zahtevkov za delo
3. Možnost pošiljanja lokacije naprave (*)
4. Možnost prikaza sprejetih opravil (*)
5. Možnost brisanja opravila s strani uporabnika, ko ga le ta zaključi
6. Možnost definiranja objekta
7. Možnost prikaza lokacije vseh uporabnikov na zemljevidu v realnem času
8. Možnost prikaza vseh objektov in njihovega stanja na zemljevidu in v seznamu (*)
9. Možnost sprožitve alarma na določenem objektu
10. Možnost ročnega izničenja alarma na objektu
11. Možnost ogleda, od koga je bilo opravilo sprejeto v realnem času (*)
12. Osveževanje stanja objektov v realnem času (*)
13. Možnost povezave različnih naprav z različnimi operacijskimi sistemi (*)
14. Avtomatska izbira objektu najbližjega vzdrževalca z alarmom
15. Avtomatsko pošiljanje zahteve naslednjemu najbližjemu vzdrževalcu ob zavrnitvi

Spodaj so opisane tri celostne rešitve, ki se po zgoraj naštetih funkcionalnostih najbolj približajo našemu predlogu rešitve.

3.1 Tasker

Aplikacija je strogo usmerjena v nadzorovanje različnih vrst vzdrževalcev na terenu. Nadrejeni vzdrževalce lahko spremlja na vsakem koraku in jim dodeljuje naloge. V kolikor ima vzdrževalec v urniku prosto mesto, je teoretično na voljo. Vzdrževalci so tako zelo omejeni in brez možnosti samoorganizacije. [9] Strukturiran opis celovite rešitve je predstavljen v tabeli 3.1.

Implementacija aplikacije v naše podjetje bi zahtevala precejšnjo preobrazbo poslovnega procesa, poleg tega pa neučinkovito rešuje problem komunikacije. Rešitev zato ni primerna.

Funkcionalnosti:	1, 3, 6, 7, 13
Deluje na sistemih:	Android in iOS
Namizna ali spletna aplikacija:	Da
Prednosti:	▪ Ogled že prepotovanih poti vzdrževalcev, z namenom optimiziranja in nadzora ▪ Vpogled v statistiko napak v poslovnem procesu
Pomanjkljivosti:	▪ Nezmožnost avtomatskega pošiljanja zahtevka najbližjemu vzdrževalcu ▪ Nezmožnost sprejetja in zavrnitve zahtevka za delo ▪ Nezmožnost izničenja alarma s strani uporabnika ▪ Koordinator v primeru zasedenosti vzdrževalca pošiljanje opravlja ponovno ▪ Nezmožnost definiranja objekta ▪ Nezmožnost avtomatskega obvladovanja zavrnjenega zahtevka ▪ Predvideva vlogo dispečerja za poln delovni čas v podjetju

Tabela 3.1 Predstavitev informacijske rešitve Tasker

3.2 Mobilplan

Aplikacija se osredotoča na popolno upravljanje zaposlenih in ne samo na mobilno enoto. Zaposleni v aplikaciji beležijo svoje delovne ure in druga sredstva, potrebna za delo. Na podlagi teh podatkov aplikacija pripravi obračun delovnega časa. Delo znotraj aplikacije je razdeljeno v projekte. [10] Vse naštete značilnosti aplikacije, ki so prikazane tudi v tabeli 3.2, so sicer primerne za delovanje vsakega podjetja, vendar za reševanje našega problema povsem nepomembne.

Rešitev je za implementacijo v našem primeru neprimerna, saj bi bila preobrazba poslovnega procesa prevelika, če bi želeli izkoristiti večino funkcionalnosti, ki jih ponuja.

Funkcionalnosti:	1, 3, 4, 6, 7, 8, 12, 13
Deluje na sistemih:	Android in iOS
Namizna ali spletna aplikacija:	Da
Prednosti:	▪ Ogled že prepotovanih poti vzdrževalcev z namenom optimiziranja in nadzora ▪ Vpogled v statistiko napak v poslovnem procesu ▪ Beleženje časa delovanja, surovine potrebne za delovanje objekta, čas od zadnjega obiska objekta ▪ Priprava obračuna delovnega časa ▪ Komunikacija zaposlenih znotraj aplikacije ▪ Optimizacija porabe potrošnega materiala
Pomanjkljivosti:	▪ Nezmožnost avtomatskega pošiljanja zahtevka najbližjemu vzdrževalcu ▪ Nezmožnost sprejetja in zavrnitve zahtevka za delo ▪ Nezmožnost izničenja alarma s strani uporabnika ▪ Koordinator v primeru zasedenosti vzdrževalca pošiljanje opravlja ponovno ▪ Nezmožnost avtomatskega obvladovanja zavrnjenega zahtevka

Tabela 3.2 Predstavitev informacijske rešitve Mobilplan

3.3 Recursion

Aplikacija je namenjena izključno procesu obveščanja mobilne delovne enote v podjetjih, ki uporabljajo sisteme SCADA. Vloga kontaktne osebe s to aplikacijo ni več potrebna. Vodja servisne službe le nadzira in urgira, če je to potrebno. [11] Rešitev je za reševanje našega problema najbolj primerna od naštetih, saj rešuje natanko isti problem, kot smo ga opisali v prejšnjem poglavju. Kljub temu aplikaciji manjkajo nekatere izmed bistvenih funkcionalnosti. V našem primeru bi bila implementacija izvedljiva, ne pa nujno učinkovita. Podrobnosti celovite rešitve so predstavljene v tabeli 3.3.

Funkcionalnosti:	1, 2, 3, 4, 5, 6, 9, 11, 12, 13
Deluje na sistemih:	Android in iOS
Namizna ali spletna aplikacija:	Da
Prednosti:	▪ Omogoča direktno integracijo v sistem SCADA ▪ Alarm vsebuje stopnjo resnosti ▪ Možno poizvedovanje stanja naprave ▪ Vsi vpleteni vidijo, kdo je sprejel alarm ▪ Omogoča generiranje poročil ▪ Vpogled v zgodovino delovanja naprave
Pomanjkljivosti:	▪ Nezmožnost prikaza lokacije uporabnikov na zemljevidu ▪ Nezmožnost izničenja alarma s strani uporabnika ▪ Nezmožnost ročnega izničenja alarma na objektu

Tabela 3.3 Predstavitev informacijske rešitve Recursion

Pregled je pokazal, da nobena izmed zgoraj naštetih aplikacij ne vsebuje vseh ključnih funkcionalnosti. Večina rešitev zaobjema precej večji problem in so zato neprimerne za implementacijo v našem primeru, saj bi se podjetje moralo preveč prilagajati in tako celo spremeniti svoj poslovni proces, kar pa je nesprejemljivo. Najbolj pomembne funkcionalnosti izmed naštetih so: avtomatska izbira objektu najbližjega vzdrževalca z alarmom, možnost sprejetja zahtevka za delo, možnost zavrnitve zahtevka za delo, avtomatsko obvladovanje zavrnjenega zahtevka. Vse te funkcionalnosti vsebuje samo aplikacija Recursion, vendar ne vključuje drugih funkcionalnosti, ki so za učinkovito reševanje problema v podjetju Petrol d.d. nujne.

Ostale aplikacije se osredotočajo bolj na popoln nadzor nad zaposlenimi na terenu, kot na dejansko komunikacijo med kontaktno osebo v pisarni in posameznim vzdrževalcem. V okviru diplomskega dela je bil razvit predlog rešitve, ki vsebuje vse ključne funkcionalnosti.

4 Rešitev za obveščanje vzdrževalcev

V prejšnjih dveh poglavjih smo izpostavili kar nekaj pomanjkljivosti in možnih izboljšav v poslovnem procesu in obstoječih rešitvah. Te pomanjkljivosti so hkrati motiv za izdelavo in implementacijo naše rešitve. Cilj je jasen: izboljšati komunikacijo med kontakno osebo in vsemi vzdrževalci na terenu ter tako prihraniti čas.

Proces alarmiranja je v vseh primerih popolnoma enak do vključno aktivnosti obveščanja kontaktne osebe (slika 2.3) o nastalem problemu na objektu. Kontaktna oseba nato ne kliče posameznih vzdrževalcev preko mobilnega telefona in ne uporablja katerega koli drugega komunikacijskega sredstva, pač pa uporabi našo spletno aplikacijo. Spletna aplikacija v podatkovni bazi glede na izbrani objekt poišče vzdrževalca, ki se trenutno nahaja najbližje. Vzdrževalcu se pošlje zahtevek za delo. Zahtevek vsebuje kodo projekta, kateremu objekt pripada, in ime tega objekta. Koda projekta je enaka vsem objektom, ki so bili vzeti v oskrbo s toploto z enako pogodbo. Objekt je vezan na enako kodo projekta, dokler se pogodba o oskrbovanju ne izteče. Podatka, vsebovana v zahtevku, zadostujeta, da vzdrževalec natančno ve, kje se objekt nahaja. Vzdrževalec lahko zahtevek za delo

sprejme ali zavrne. V kolikor zahtevek sprejme, se mu le ta doda na seznam opravil. V obratnem primeru se zahtevek za delo pošlje vzdrževalcu, ki je po razdalji do objekta z napako drugi najbližji. Tako vzdrževalcem damo popolno možnost samoorganizacije. Po tem načinu se zahtevki za delo pošiljajo do zadnjega vzdrževalca, ki pa nima več možnosti zavrniti zahtevek. Lahko ga le sprejme. Poleg dobi še navodilo, naj pokliče v pisarno, v kolikor zahtevka ne more sprejeti. Če zahtevek za delo zavrnejo vsi vzdrževalci, to pomeni, da so vzdrževalci trenutno preobremenjeni. Ko vzdrževalec alarm na objektu odpravi, s klikom na opravilo le tega izbriše. Kontaktna oseba lahko v realnem času nadzoruje spremembe v spletni aplikaciji.

4.1 Pregled dela

Začeli smo s preučitvijo konkretnega poslovnega procesa v podjetju. V sklopu te dejavnosti smo se večkrat sestali z vodjo servisne službe in opazovali posamezne vzdrževalce pri delu. Poleg tega smo dodobra raziskali strojno in programsko opremo na mobilnih telefonih vzdrževalcev. S temi podatki smo začeli z načrtovanjem same mobilne in spletne aplikacije. Sem, poleg celostne grafične podobe, spada tudi razporeditev elementov na vsako okno aplikacije posebej, z namenom doseganja enostavne in učinkovite uporabe (*angl. wireframe*), in izdelava hierarhije oken aplikacij za boljše razumevanje delovanja (*angl. site map*).

Sledila je izbira primerne platforme, ki bo hranila in servirala podatke. Odločali smo se na osnovi kriterijev: enostavnost uporabe, namembnost, ponudba storitev in cena. Pogojem so ustrezale platforme: Kaa, Carriots, ThingsWorx in Google Firebase. Po podrobnem pregledu vsake izmed teh smo izbrali slednjo. Poleg tega, da Google Firebase zadostuje vsem našim potrebam uporabe, sta operacijski sistem, na katerem bo tekla naša aplikacija, in razvojno okolje prav tako produkta podjetja Google. Podpora sistema Firebase je zato odlična.

Z izdelanim načrtom in izbrano platformo smo pričeli s pisanjem kode. Najprej je bila na vrsti mobilna aplikacija, za katero smo potrebovali večino namenjenega časa. Sledila je priprava platforme Google Firebase. Na koncu pa smo se lotili še uporabniškega vmesnika oziroma spletne aplikacije, namenjene kontaktni osebi. Po zaključku imple-

mentiranja aplikacij smo v rešitev vnesli resnične podatke in jo preizkusili na dejanskem primeru v testnem podjetju.

Podjetje, kjer smo testirali našo rešitev, za podporo svoji dejavnosti uporablja sistem SCADA. Poznavanje osnov sistema SCADA je ključno, da razumemo poslovni proces, ki ga s to nalogo izboljšujemo. Rešitev je z nekaj dopolnitvami pripravljena na integracijo s sistemom SCADA, vendar je bilo to v času testiranja nemogoče udejaniti zaradi varnostne politike podjetja. Za integracijo rešitve in sistema SCADA bi morali imeti dostop do prejemanja podrobnosti o alarmih v realnem času. V podjetju Petrol d.d. v ta namen uporabljajo protokol OPC AE (*angl. Open Platform Communications Alarm Event*). Protokol, kot dodatek sistemu SCADA, omogoča nadzor željenih naprav in informiranje odjemalcev o stanju naprav. [12] Na naši strani bi bilo potrebno razviti storitev, s pomočjo katere bi se naročili na prejemanje podatkov o alarmih. Storitev bi podatke prejela, jih preoblikovala v željeno obliko ter jih posredovala v spletno platformo. Na ta način nebi bilo potrebno alarma sprožiti ročno v spletni aplikaciji, pač pa bi se to zgodilo samodejno. Spletna aplikacija, bi bila uporabljena samo za prikaz podatkov. Pri tem mobilne aplikacije ne bi spreminjali. Spletno platformo pa bi morali prilagoditi sprejemanju podatkov s strani storitve.

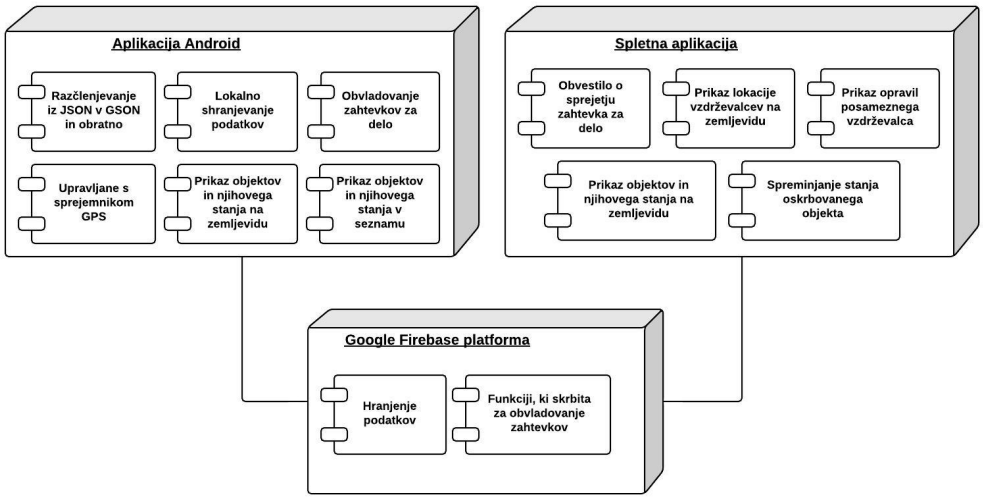
Diagram na sliki 4.1 prikazuje komponente, razvite v tej nalogi. Vsaka izmed glavnih komponent in vsebovanih komponent je opisana v naslednjih poglavjih.

4.2 Platforma Google Firebase

Platformo Google Firebase smo v projektu uporabili za hranjenje podatkov v podatkovni bazi, izvajanje funkcij v oblaku, pošiljanje zahtevkov za delo na primerno napravo in obvladovanje POST zahtevkov.

Izbrana platforma v prvi vrsti ponuja podatkovno bazo z možnostjo sinhroniziranja v realnem času. Gre za podatkovno bazo tipa NoSQL. Platforma ponuja možnost razširitve aplikacij s funkcijami v oblaku. Omogoča nam tudi podrobno analizo delovanja aplikacije in izpadov aplikacije iz normalnega delovanja, pošiljanje sporočil uporabnikom v realnem času, avtentikacijo uporabnikov na več načinov, shranjevanje večjih datotek, itd. Z upo-

rabo platforme Googel Firebase nam ni treba skrbeti za urejenost podatkov v oblaku niti za varnostne kopije. Podpira razvijanje v programskih jezikih: C++, Java, Javascript, PHP, Python in Swift.



Slika 4.1 Arhitektura celovite rešitve

Google Firebase omogoča uporabnikom uporabo zastonj, v kolikor ne presežemo števila kapacitet, ki so določene za vsako razširitev znotraj platforme. Za aplikacije, ki predvidevajo večje število uporabnikov ali bi zaradi kakršnega koli drugega razloga presegle osnovne kapacitete, lahko izberemo enega izmed dveh stroškovnih planov. [13]

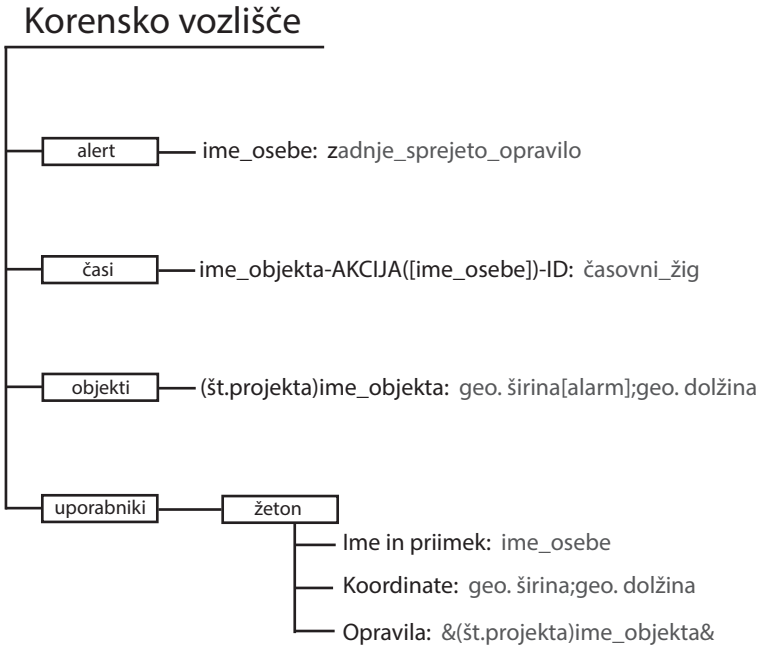
4.2.1 Hranjenje podatkov

Podatki se v spletni platformi Google Firebase hranijo v obliki JSON (*angl. JavaScript Object Notation*). JSON je besedilni format, namenjen podatkovni izmenjavi. Odlikuje ga enostavnost razumevanja, uporabe in ustvarjanja s strani ljudi ter računalnikov. Osnova za delovanje omenjenega formata je programski jezik JavaScipt, od katerega pa ni odvisen. Podatki so v zapisu ključ: vrednost. [14]

Podatki se hranijo v drevesni strukturi. Za delovanje aplikacije potrebujemo podatke o uporabnikih in objektih. Poleg tega hranimo še časovne žige ob določenih dogodkih v obravnavi alarma in informacijo, kdo je sprejel aktivni alarm.

Uporabnike med seboj razlikujemo z nizom dolgim 152 znakov, ki mu rečemo žeton. Žeton je potreben, da platforma ve, kam poslati zahtevek za delo. Mobilna aplikacija s pomočjo določene knjižnice ob prvi interakciji ustvari žeton in ga zapiše v bazo. Žeton služi kot korensko vozlišče za vsakega uporabnika. Podatki, ki jih hranimo za vsakega uporabnika, so: ime, koordinate, sprejeta opravila.

V podatkovni bazi hranimo podatke objektov, ki so potrebni, da vzdrževalec razbere lokacijo objekta in da aplikacija lahko objekt prikaže na zemljevidu. Ti podatki vključujejo številko projekta, kateremu objekt pripada, naslov oziroma ime objekta ter točno lokacijo v obliki koordinat. Struktura podatkovne baze je prikazana na sliki 4.2.



Slika 4.2 Prikazuje strukturo podatkovne baze, ki jo uporabljamo v aplikaciji na platformi Google Firebase

Za vsak alarm hranimo najmanj tri časovne žige. Časovni žig se zapiše ob naslednjih akcijah: ob sprožitvi alarma s strani kontaktne osebe v pisarni, ob sprejetju ali zavrnitvi alarma s strani vzdrževalca in ob zaključku alarma, ko vzdrževalec odpravi napako na napravi. V primeru da se enkrat ali večkrat zgodi, da je alarm zavrnjen, je v bazi tudi primerno več zapisov enega alarma. Zapis v podatkovni bazi ne vsebuje samo časovnega žiga, pač pa tudi osebo, akcijo in ime objekta, na katerem se alarm izvaja.

Pod korenskim vozliščem z imenom *alert* hranimo informacijo o tem, kdo je sprejel zadnji alarm z namenom prikaza te informacije v spletni aplikaciji v realnem času.

4.2.2 Funkciji v oblaku

Za pošiljanje zahtevkov za delo najbližjemu vzdrževalcu in za obvladanje zavrnjenega zahtevka za delo skrbita dve funkciji. Razvili smo ju v programskem jeziku JavaScript. Omenjeni programski jezik dovoljuje implementacijo kompleksnejših funkcij v spletne strani. Omogoča dinamično posodabljanje vsebine, uporabo multimedijske vsebine, animacijo ter mnogo več. [15] Najprej se osredotočimo na funkcijo, ki skrbi za računanje vrste objektu najbližjih vzdrževalcev z alarmom in pošiljanje zahtevka za delo najbližjemu izmed njih.

Funkcija *izracunajPoslji* se zažene ob spremembi vrednosti kateregakoli izmed objektov. Ob sprožitvi alarma s strani kontaktne osebe se v bazi spremeni vrednost objekta z alarmom. Funkcija iz baze razbere, na katerem objektu je bil sprožen alarm, ter si zapomni geografsko širino in dolžino tega objekta. S pomočjo shranjenih koordinat objekta izračunamo razdaljo od objekta do posamezngea vzdrževalca posebej ter vrednosti shranimo v seznam z imenom *prvi*. Na enakem mestu se v seznamu z imenom *drugi* nahaja žetom vzdrževalca. Seznam *prvi* uredimo v naraščajočem vrstnem redu ter temu primerno prilagodimo seznam *drugi*. Iz seznama *drugi* izberemo žeton najbližjega vzdrževalca, ki ga najdemo na začetku seznama. Izbrani žeton nam služi kot naslov za pošiljanje zahtevka za delo na mobilno napravo vzdrževalca. Zahtevek za delo vsebuje poleg imena objekta z alarmom še seznam *drugi*, ki hrani vse ostale žetone vzdrževalcev, urejene v naraščajočem vrstnem redu po oddaljenosti od objekta z alarmom v metrih. Slednji podatek je za prejemnika zahtevka transparenten. Implementacijo pošiljanja zahtevka prikazuje primer izvorne kode 4.1.

```
const paket = {
data: {
title: 'Alarm na objektu!',
body: request.body.objekt + "$"+ ostali.toString()
}
```

```
};

admin.messaging().sendToDevice(oseba, payload);
```

Primer izvorne kode 4.1 Pošiljanje zahtevka za delo

Za izračun razdalj med objektom in vzdrževalci smo uporabili haversinovo formulo. Formula izračuna najkrajšo razdaljo med dvema točkama na sferičnem objektu ne glede na ovire med točkama. [16] V našem primeru je to dovolj, saj so vzdrževalci razpršeni po terenu v precej velikih razdaljah. Formula je prikazana v primeru izvorne kode 4.2.

```
function izracunRazdalje(lat1,lon1,lat2,lon2) {
var R = 6371;
var dLat = deg2rad(lat2-lat1);
var dLon = deg2rad(lon2-lon1);
var a = Math.sin(dLat/2) * Math.sin(dLat/2) + Math.cos(deg2rad(lat1)) *
    Math.cos(deg2rad(lat2)) * Math.sin(dLon/2) * Math.sin(dLon/2);
var c = 2 * Math.atan2(Math.sqrt(a), Math.sqrt(1-a));
var d = R * c;

return d;
}

function deg2rad(deg) {
return deg * (Math.PI/180)
}
```

Primer izvorne kode 4.2 Izračun razdalje med dvema paroma koordinat [16]

Če se uporabnik ob prejetju zahtevka za delo odloči, da le tega trenutno ne more sprejeti, ga zavrne. Zahtevek za delo mobilna naprava vzdrževalca pošlje nazaj v spletno platformo. Zavrnjeni zahtevek vsebuje ime objekta z alarmom in seznam žetonov vseh vzdrževalcev, urejenih v naraščajočem vrstnem redu, s tem da je odstranjen žeton najbližjega vzdrževalca, torej tega, ki ga zavrne. Za obvladovanje zavrnjenih zahtevkov na platformi skrbi funkcija *izračunajPoslji2*. Ob prejetju zavrnjenega zahtevka funkcija zahtevek pošlje mobilni napravi z žetonom, ki se nahaja na začetku seznama. V kolikor

je v seznamu ostal samo še en žeton, torej je v vrsti samo še zadnji, najbolj oddaljen vzdrževalec, se pošlje zahtevek za delo, katerega ni možno zavrniti. V primeru da tudi ta vzdrževalec ne more sprejetu opravila, v zahtevku za delo dobi navodila, da naj pokliče v pisarno.

4.3 Spletna aplikacija

Aplikacija je namenjena kontaktni osebi v podjetju. Omogoča vpogled v stanje objektov in spremljanje lokacije vzdrževalcev v realnem času. Razvili smo jo v razvojnem okolju InteliJ v programskemu jeziku JavaScript v kombinaciji s HTML (*angl. Hyper Text Markup Language*) jezikom. HTML je standardni jezik za oblikovanje spletnih strani. Z različnim oznakami opisuje celotno spletno stran. [17] Razvojno okolje InteliJ je razvito s strani podjetja jetBrains. Prvotni programski jezik je Java. Obstaja v plačljivi in zastonjski različici. [18]

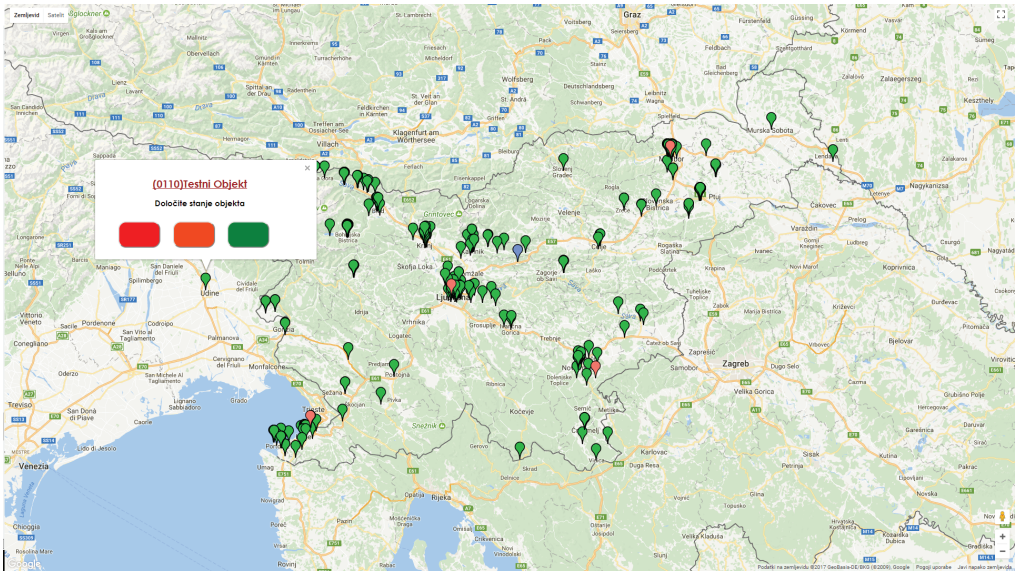
Spletna aplikacija prikazuje Google zemljevid, osredotočen na Slovenijo, na katerem se na različnih lokacijah nahajajo objekti. Potrebne podatke za prikaz objektov ob zagonu aplikacije preberemo iz podatkovne baze. Objekt je prikazan s točko zelene, oranžne ali rdeče barve, odvisno od njegovega stanja. Zelena barva pomeni, da objekt deluje brezhibno, oranžna, da je na objektu manjša napaka, ponavadi ne nujna, in rdeča, ki pomeni, da objekt nujno potrebuje pozornost. Kontaktna oseba v podjetju lahko stanje objekta spreminja s klikom na točko in tako sproži alarm ali pa ga izniči. Na zemljevidu so z modro točko, poleg vseh objektov, prikazani tudi vzdrževalci. Njihova lokacija se osvežuje v realnem času. Ob kliku na točko vzdrževalca lahko vidimo njegovo ime in opravila, ki jih je sprejel. Spletna aplikacija nam sporoči, kdo od vzdrževalcev je sprejel določen zahtevek za delo.

Aplikacija ne teče na spletu, temveč lokalno, v podjetju. Razlog za tako odločitev je varovanje podatkov testnega podjetja in njihovih zaposlenih. Spletna aplikacija je prikazana na sliki 4.3.

4.3.1 Spreminjanje stanja objekta

Spreminjanje stanja objekta smo implementirali na preprost način. Ob kliku na objekt se nam odpre manjše podokno, v katerem lahko kliknemo enega izmed treh gumbov —

glede na resnost alarma.



Slika 4.3 Prikazuje spletno aplikacijo z nekaj aktivnimi alarmi, oknom za sprožitev alarma ter vzdrževalcem blizu kraja Trojane

V kolikor ob kliku v podoknu izberemo drugo barvo, ki trenutno na zemljevidu predstavlja objekt, se stanje objekta spremeni in sproži se alarm. Stanje na določenem objektu je v podatkovni bazi označeno z znakom ! za rdeče stanje, z znakom * za oranžno stanje in brez znaka za zeleno stanje. Znak je zapisan za prvo koordinato objekta, torej za geografsko širino. Primer zapisa objekta z oranžnim alarmom: *(0110)Testni Objekt: 46.080731*;13.214766.*

Ob izbiri želenega stanja se ustvari povezava do izbranega objekta v podatkovni bazi. Na splošno nam povezava služi, da preko nje pridobimo želene podatke, jih po potrebi spremenimo ter jih zapišemo nazaj na isto mesto. V tem primeru podatke preko povezave samo zapisujemo. Podatke o koordinatah pridobimo iz že obstoječe točke, ki predstavlja objekt, informacijo o izbranem stanju pa nam poda enden od treh gumbov, ki je bil uporabljen za izbiro. Glede na izbrano stanje prvi koordinati na koncu dodamo primeren znak in podatek zapišemo v podatkovno bazo.

Stanje objekta lahko spremenijo tudi vdrževalci. Ko alarm na objektu odpravijo, v mobilni aplikaciji to tudi potrdijo in barva objekta se spremeni v zeleno. Da so stanja

objektov med mobilnimi napravami in spletno aplikacijo sinhronizirana v realnem času, v spletni aplikacij konstantno poizvedujemo, če se je v podatkovni bazi na kateremkoli objektu zgodila kakšna sprememba. Način konstantnega poizvedovanja prikazuje primer izvorne kode 4.3. V kolikor se sprememba zgodi, določimo, v kakšnem stanju je bil objekt pred spremembo, in znak, ki določuje alarm, zamenjamo s praznim nizom.

```
var ref = firebase.database().ref("...");
ref.on("child_changed", function (snapshot) {
// .......
// .......
});
```

Primer izvorne kode 4.3 Konstantno poizvedovanje za spremembami v podatkovni bazi

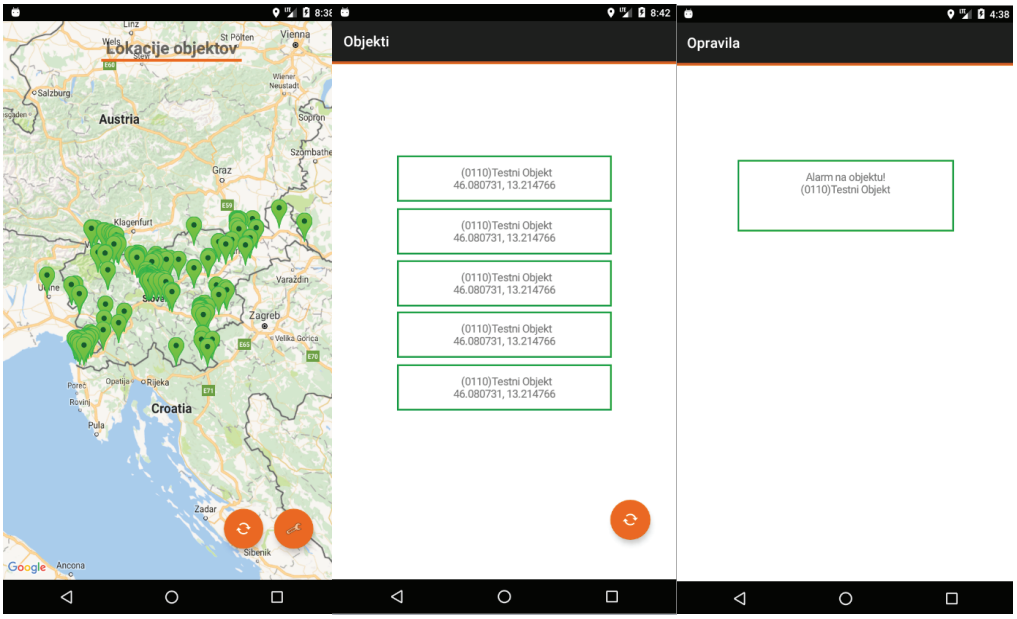
4.3.2 Obvestilo o sprejemu zahtevka

Ob sprožitvi alarma se le ta pojavi v obliki zahtevka na mobilni napravi najbližjega vzdrževalca. Vzdrževalec lahko zahtevek zavrne ali sprejme. V kolikor vzdrževalec sprejme zahtevek, se podatek o tem, da je ta dotični vzdrževalec sprejel zahtevek, zapiše v podatkovno bazo kot vrednost pod korensko vozlišče *alert*. Ključ je v tem primeru ime vzdrževalca. Potrdilo o sprejemu zahtevka za delo se v spletni aplikaciji pokaže kot obvestilo, ki ga mora kontaktna oseba potrditi. Obvestilo v spletno aplikacijo implementiramo s stavkom *window.alert*. Da se obvestilo pokaže v realnem času, storimo enako kot pri spreminjanju stanja objekta. Konstantno poizvedujemo o spremembah zapisa podatkov v podatkovni bazi pod prej omenjenim korenskim vozliščem.

4.4 Mobilna aplikacija

Mobilna aplikacija je razvita za mobilne naprave z operacijskim sistemom Android različice 6.0 ali novejše. Android je odprtokodni operacijski sistem, primarno namenjen za mobilne naprave z zaslonom na dotik, ki ne vsebujejo veliko pomnilnika in imajo počasnejši procesor. Operacijski sistem Android se med drugim pojavlja tudi na televevizijskih sistemih, pametnih urah, igralnih konzolah, prenosnih računalnikih in digitalnih kamerah. Primarno je operacijski sistem razvijalo podjetje Android, vendar od leta 2005 deluje pod okriljem podjetja Google. [19] Za razvoj smo uporabili razvojno okolje Android Studio v

kombinaciji s programskim jezikom Java. Android Studio je uradno razvojno okolje za razvoj aplikacij, ki tečejo na operacijskem sistemu Android. Razvijajo ga pri podjetju Google. Leta 2013 je ugledal luč sveta. Kot podlago za razvoj te programske opreme so uporabili JetBrains IntelliJ IDEA. Android Studio je brezplačen in na voljo za večino operacijskih sistemov. [20]



Slika 4.4 Prikazuje glavna okna mobilne aplikacije: lokacijo in stanje objektov na zemljevidu, lokacijo in stanje objekta v seznamu, sprejeta opravila.

Aplikacija za učinkovito delovanje in omogočanje vseh funkcionalnosti potrebuje internetno povezavo in vključen GPS (*angl. Global Positioning System*) sprejemnik. Ob vzpostavitvi aplikacije je nujno, da uporabnik vnese svoje ime, ki služi za razlikovanje vzdrževalcev ob prikazu ne zemlejvidu v spletni aplikaciji. Ob potrditvi imena se v bazo poleg imena zapišejo tudi njegove trenutne koordinate. Na domačem zaslonu aplikacje lahko spremljamo stanja vseh objektov. Objekti so prikazani enako kot v spletni aplikaciji. Ob kliku na objekt vidimo številko projekta, kateremu pripada, in njegovo ime. Na domačem zaslonu najdemo gumb za osveževanje podatkov. Osveževanje v mobilni aplikaciji se namreč ne odvija avtomatsko. Razlog za to je poraba baterije. Ko uporabnik želi videti osvežene podatke, pritisne omenjeni gumb. Vsi podatki o objektu se prav tako nahajajo v podoknu aplikacije v obliki seznama. Poleg podokna s seznamom objektov v aplikaciji najdemo še podokno s seznamom opravil, podokno, kjer uporabnk

vpiše svoje ime, ter podokno z informacijami o aplikacji. Glavna okna mobilne aplikacije so prikazana na sliki 4.4.

Za lokalno shranjevanje podatkov v mobilni napravi smo uporabili *Shared Preferences.* Omenjeni način shranjevanja deluje po principu parov ključ, vrednost. V *Shared Preferences* lahko shranjujemo samo nize. Tega načina se ne uporablja pri shranjevanju relacij med podatki in bolj kompleksnimi objekti. Shranjeni podatki bodo ostali v napravi, dokler aplikacije ne odstranimo. Podatki so dostopni iz vseh aktivnosti aplikacije, lahko pa dostopnost tudi omejimo. [21] V *Shared Preferences* smo shranili vse objekte in njihova stanja, sprejeta opravila vzdrževalca ter ime vzdrževalca. Za shranjevanje in pridobivanje podatkov uporabimo razred *SharedPreferences.* Osnovno delovanje omenjenega načina shranjevanja podatkov je prikazano v primeru izvorne kode 4.4.

```
//zapisovanje
SharedPreferences sharedPrefs =
    PreferenceManager.getDefaultSharedPreferences(...);
SharedPreferences.Editor editor = sharedPrefs.edit();
editor.putString(kljuc, vrednost);
editor.commit();

//pridobivanje
SharedPreferences sharedPrefs =
    PreferenceManager.getDefaultSharedPreferences(...);
String s = sharedPrefs.getString(kljuc, privzeta_vrednost);
```

Primer izvorne kode 4.4 Uporaba razreda *SharedPreferences* za lokalno shranjevanje podatkov

4.4.1 Povezava na Google Firebase

Preko mobilne aplikacije se na platformo primarno povezujemo zaradi pridobivanja in zapisovanja ažurnih podatkov o objektih. Povezava s platformo igra pomembno vlogo tudi pri prejemanju zahtevkov za delo, sporočanju odgovora na zahtevek ter sinhronizaciji lokacije vzdrževalca med mobilno in spletno aplikacijo, kjer je povezava tudi najbolj frekventno uporabljena. Podatke o objektih najprej pridobimo ob zagonu aplikacije in nato vsakič, ko uporabnik želi osvežene podatke. Ko podatke pridobimo, jih najprej zložimo

v seznam, ki pa ga je potrebno pretvoriti v Java tip *String*, v kolikor ga v napravi želimo shraniti lokalno. Za pretvorbo smo uporabili knjižnico GSON (*angl. Google Script Object Notation*). Omenjena knjižnica se uporablja za pretvorbo Java objektov v JSON obliko in obratno. Knjižnico odlikuje enostavna uporaba, prav tako pa je odprtokodna. [22] Pretvorba z uporabo knjižnice se uporablja samo v primeru pridobivanja podatkov o objekih. Primer izvorne kode 4.5 prikazuje pridobitev vseh objektov iz podatkovne baze ob zagonu aplikacije in uporabo knjižnice GSON za shranjevanje le teh v lokalno shrambo.

```
FirebaseDatabase database = FirebaseDatabase.getInstance();
DatabaseReference myRef = database.getReference("objekti");
ArrayList<String[]> objekti = new ArrayList<>();

myRef.addListenerForSingleValueEvent(new ValueEventListener() {

@Override
public void onDataChange(DataSnapshot dataSnapshot) {
for (DataSnapshot child : dataSnapshot.getChildren()) {
// ...
objekti.add(child);
}

SharedPreferences sharedPrefs =
    PreferenceManager.getDefaultSharedPreferences(...);
SharedPreferences.Editor editor = sharedPrefs.edit();
Gson gson = new Gson();
String json = gson.toJson(objekti);
editor.putString("objekti", json);
editor.commit();
}
```

Primer izvorne kode 4.5 Pridobitev objekov s podatkovne baze in shranjevanje le teh v napravo

4.4.2 Obvladovanje zahtevkov za delo

Mobilna naprava prejme zahtevek za delo ob sprožitvi alarma nad objektom v spletni aplikaciji. Kateri izmed vzdrževalcev prejme zahtevek, se določi glede na rezultat izračuna

haversinove formule. Za prikaz podatkov zahtevka v mobilni napravi uporabimo javanski razred, *MyFirebaseMessagingService*, ki je razširitev razreda *FirebaseMessagingService*. Metoda *onMessageReceived* se sproži, ko napravo doseže zahtevek za delo. V metodi iz zahtevka izluščimo podatke, jih lokalno shranimo ter preko razreda *Intent* odpremo aktivnost, ki zahtevek za delo prikaže in ponudi opcijo sprejema ali zavrnitve. V kolikor uporabnik med prejemom zahtevka za delo aplikacije ne uporablja, se pokaže obvestilo o zahtevku za delo v opravni vrstici.

V primeru zavrnjenega zahtevka se le ta pošlje z uporabo POST zahtevka nazaj v platformo. Omenjeni zahtevek se uporablja za pošiljanje občutljivih informacij preko omrežij. POST zahtevek prav tako nima omejitve pri količni podatkov ter podpira pošiljanje binarnih podatkov. [23] Pri pošiljanju POST zahtevkov smo šumnike zamenjali z znaki angleške abecede, da smo se izognili težavam. Ob prejemu zahtevka s strani naslednje naprave smo znake, ki med pošiljanjem predstavljajo šumnike, zamenjali nazaj, tako da uporabnik ni prizadet.

4.4.3 Pošiljanje lokacije

Kontaktna oseba lahko v spletni aplikaciji spremlja lokacijo posameznega vzdrževalca. Natančna lokacija ni pomembna le za spremljanje vzdrževalcev, pač pa tudi zato, da se ob alarmu pravilno izračuna oddaljenost vzdrževalcev. Ob zagonu mobilne aplikacije nas aplikacija preusmeri na okno z nastavitvami GPS sprejemnika, v kolikor je izključen. Prvič se koordinate pošljejo ob potrdivi imena v nastavitvah aplikacije. Takrat se tudi pojavi točka v spletni aplikaciji, ki predstavlja vzdrževalca. Frekvenca pošiljanja podatkov o lokaciji je odvisna od hitrosti premikanja vzdrževalca. Lokacija se namreč osveži, če se vzdrževalec premakne vsaj dvajset metrov zračne razdalje od prej shranjene točke. V primeru izvorne kode 4.6 je prikazano pošiljanje lokacije v spletno platformo.

```
locationManager = (LocationManager) getSystemService(LOCATION_SERVICE);
// ...
locationListener = new LocationListener() {
@Override
public void onLocationChanged(Location location) {
SharedPreferences sharedPrefs =
    PreferenceManager.getDefaultSharedPreferences(...);
```

```
String ime = sharedPrefs.getString("ime", null);
String token = FirebaseInstanceId.getInstance().getToken();
ZapisiBaza zapis = new ZapisiBaza();
double lat = location.getLatitude();
double lon = location.getLongitude();

zapis.write(token, lat, lon, ime);
}
//...
locationManager.requestLocationUpdates("gps", 0, 20, locationListener);
```

Primer izvorne kode 4.6 Pošiljanje lokacije glede na premikanje vzdrževalca

Tako za uporabo GPS sprejemnika kot tudi za uporabo omrežnega sprejemnika na mobilni napravi potrebujemo dovoljenje uporabnika. Ob prvem zagonu aplikacije se uporabniku prikaže obvestilo o uporabi omenjenih sprejemnikov. Le tega lahko potrdi ali zavrne. Vsa dovoljenja, ki jih potrebujemo za delovanje aplikacije, so zapisana znotraj strukture mobilne aplikacije, v datoteki *AndroidManifest.xml*.

5 Preizkus rešitve v podjetju

Preizkus rešitve obsega spoznavanje poslovnega procesa testnega podjetja, implementacijo in postavitev konkretne rešitve v testno podjetje in merjenje učinkovitosti dela z uporabo naše rešitve v primerjavi z obstoječim načinom dela v testnem podjetju. Največ uporabnih informacij in pomoči pri izvedbi smo pridobili s sodelovanjem in opazovanjem zaposlenih na različih nivojih poslovnega procesa ter iz podatkov, ki smo jih pridobili z dovoljenjem vodje informacijskega oddleka.

5.1 Testno podjetje

Podjetje Petrol d.d je največja energetska družba v Sloveniji po prihodkih in številu zaposlenih. Glavna dejavnost družbe je trgovanje z naftnimi derivati, plinom in ostalimi energenti. Poleg naftnih derivatov, plina in energetskih rešitev družba trži tudi vodo. [24]

Svoj predlog rešitve smo preizkusili v oddelku podjetja, Petrol d.d. Organizacijska enota energetske rešitve Bled. Oddelek na Bledu se ukvarja z različnimi oblikami energetskega pogodbeništva, prenavljanjem sistemov ogrevanja ter z gradnjo in upravljanjem energet-

sko učinkovitih sistemov daljinskega ogrevanja in vodovodnih sistemov. Rešitev obsega implementacijo strokovnih in nadzornih sistemov za daljinski nadzor in upravljanje z infrastukturo, ki jo nadzorujejo 24 ur na dan. Skrbijo za pitno, sanitarno in tehnološko vodo ter prečiščevanje le-te. [25] V okviru naše rešitve obravnavamo samo objekte, katere blejski oddelek oskrbuje z ogrevanjem. Teh je po Sloveniji nekaj več kot 500.

Oskrbovani objekt je v našem primeru stavba ali del stavbe. Vsak od objektov je priključen na toplotni vir, ki je lahko skupen za več objektov. Oskrbovani objekt lahko vsebuje več opcijskih naprav, priključenih v sistem SCADA. Število naprav je odvisno od velikosti objekta in zahtev stranke. Glavna naprava, ki je vedno prisotna, je toplotna črpalka. Na vsako od naprav je priključen PLC krmilnik. Krmilniki so si med seboj različni. V testnem podjetju trenutno največ uporabljajo PLC krmilnik z oznako TP-08. Omenjeni krmilnik ima možnost podpore štirih ogrevalnih krogov, komunikacije preko protokolov Modbus TPC in TCP/IP, vsebuje vhod za spominsko kartico in terminal za upravljanje. [26]

5.1.1 Komunikacije med objektom in sistemom SCADA

PLC krmilnik je priključen na usmerjevalnik, ki omogoča prenos podatkov preko žičnega ali brezžičnega omrežja. Testno podjetje uporablja usmerjevalnike MikroTik. Na lokacijah kjer je možna žična povezava, se uporabljajo usmerjevalniki brez GSM (*angl. Global System for Mobile communications*) sprejemnika, kjer pa te možnosti ni, pa je uporaba usmerjevalnika z GSM sprejemnikom obvezna. Žična povezava je bolj zanesljiva od mobilne in jo zato prakticirajo, kjer je le možno. Na bolj odmaknjenih lokacijah možnosti žične povezave enostavno ni, zato ni druge možnosti kot implementacija usmerjevalnika z GSM sprejemnikom.

V kolikor je žična povezava na voljo, se vprašamo o ekonomičnosti in pogojih uporabe. Žična povezava mora biti dostopna v bližni PLC krmilnika znotraj objekta, da je za naš primer uporabna. Če je v objektu žična povezava na voljo, vendar ni v bližini krmilnika, je treba preučiti ceno omogočanja žične povezave v bližino krmilnika. V primeru da žična povezava je na voljo v bližini krmilnika in jo lahko uporabimo brez dodatnih stroškov, moramo natančno določiti pogoje uporabe žične povezave s stranko. Internetna povezava je v našem primeru ključnega pomena, zato se med podjetjem in stranko ob podpisu

pogodbe sklene dogovor, ki te pogoje natančno opredeljuje. Zgodi se, da pogoji uporabe žične povezave ne zadostujejo naši uporabi ali pa je cena omogočanja žične povezave previsoka, tako kljub možnosti žične povezave implementiramo usmerjevalnik z GSM sprejemnikom. Prednost implementacije usmerjevalnika z GSM sprjemnikom je neodvisnost od strankine infrastrukture in pogojev uporabe, glavna slabost pa je, da mobilni ponudnik ne zagotavlja konstantne povezave, in zato komunikacija med usmerejvalniki in sistemom SCADA lahko za določen čas odpove. Usmerjevalnike z GSM sprejemnikom ali brez je treba primerno konfigurirati, da na lokaciji delujejo tako kot želimo.

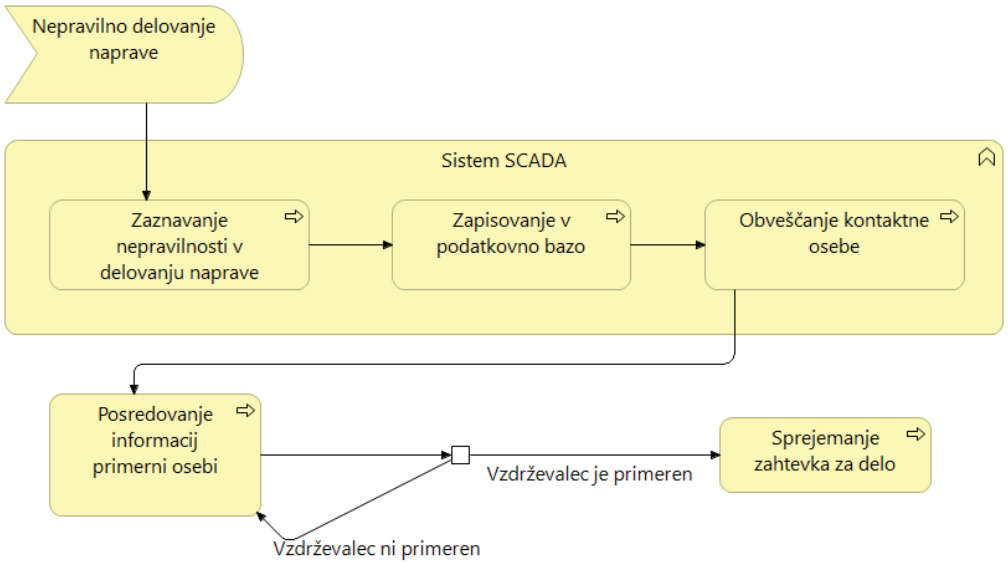
5.1.2 Servisna služba

Naloga servisne službe je odpravljanje nepravilnosti na napravah, ki se zgodijo med delovanjem. V servisni službi na Bledu, ki skrbi za ogrevanje objektov, deluje sedem vzdrževalcev in vodja servisne službe. Med sedmimi vzdrževalci sta dva dežurna. Glavna naloga vodje servisne službe je razpolaganje nalog vzdrževalcem. Poleg tega se vodja udeležuje sestankov znotraj podjetja in je v kontaktu s strankami. V primeru da je vodja zaseden z drugimi obremenitvami, nalogo razdeljevanja obveznosti prevzame eden od dežurnih vzdrževalcev. Vzdrževalci so med delovnim časom razporejeni po terenu in opravljalo zadane naloge, medtem pa so lahko s klicem na mobilni telefon obveščeni o novem alarmu na objektu, ki je hkrati zahtevek za delo. Celoten kolektiv servisne službe se enkrat na teden sestane in pogovori o dogajanju na terenu. Med zaposlenimi v servisni službi obstaja dežurni mobilni telefon, ki ga ima praviloma kontaktna oseba. Uporablja se za obveščevanje o alaramih.

5.1.3 Proces klasičnega obveščanja vzdrževalcev

Nepravilno delovanje ogrevalnih sistemov je največkrat rezultat izpada elektrike, napačne konfiguracije toplotne črpalke, zatajitve komponent v toplotni črpalki ali ena od mnogih drugih nepredvidljivih situacilj. Alarm se sproži kot rezultat enega od dogodkov: nepravilno delovanje naprave na objektu, če program, ki teče v PLC krmilniku, kljub pravilnem delovanju naprave zazna padec ali rast vrednosti izven določenih meja, nezadovoljna stranka preko e-pošte ali telefona sproči posledico napačnega delovanja. V kolikor je bil alarm zaznan s strani PLC krmilnika, informacija do kontaktne osebe pride preko e-poštnega sporočila na e-poštni naslov vodje in SMS spročila na dežurno mobilno številko.

Podatki se med PCL krmilniki in centralnim računalnikom v podjetju prenašajo preko protokola Modbus TCP ali TCP/IP, odvisno od vrste krmilnika, ki je uporabljen. Centralni računalnik skrbi za zapisovanje podatkov v podatkovno bazo in posredovanje informacij do kontaktne osebe. Ko kontaktna oseba prejme informacijo o nastalem alarmu na določeni lokaciji, le to posreduje naprej do prostega vzdrževalca, ki je v razumljivem dometu objekta z napako. Na tem mestu se pojavi problem in hkrati velika možnost izboljašave, ki jo v tej doplomski nalogi tudi naslavljamo.



Slika 5.1 Diagram prikazuje aktivnosti poslovnega procesa v primeru klasičnega obveščanja vzdrževalcev v podjetju Petrol d.d.

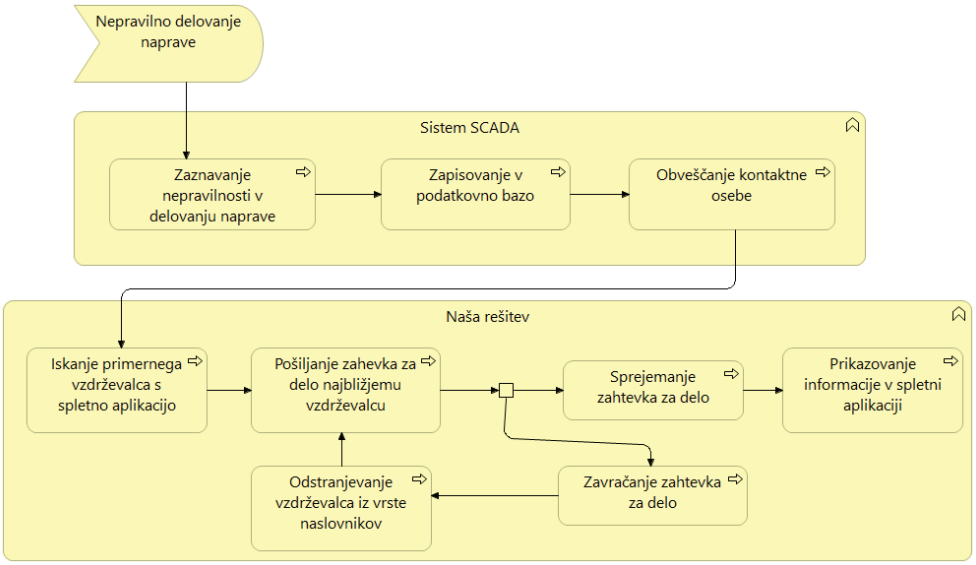
Kontaktna oseba, pa naj bo to dežurni vzdrževalec ali vodja servisne službe, nima na voljo podatkov o trenutnem delu vdrževalca niti o tem, kje se nahaja. Kontaktna oseba preko mobilnega telefona s klicanjem pozkuša najti primernega vzdrževalca za razrešitev alarma, kar pa je lahko kar zamudna in zahtevna naloga. Pokliče vzdrževalca, ki je v tistem trenutku po njegovem mnenju najbolj primeren za delo. Velikokrat se zgodi, da je vzdrževalec v trenutku klica enostavno preveč zaposlen, da bi se oglasil na mobilni telefon in sprejel še eno obveznost ali pa je zaradi drugega alarma preveč oddaljen in zato neprimeren za razreševanje alarma. Tako kontaktna oseba opravi veliko nepotrebnih klicev. Posledica pa je lahko kljub temu da kontaktna oseba opravi samo en klic, ne optimalno

izbran vzdrževalec glede na njegovo lokacijo in trenutno zaposlitev, saj nima podatkov o delovanju drugih vzrževalcev v tem času. Vzdrževalci v testnem podjetju delujejo po principu samoorganizacije in tega z uvedbo naše rešitve ne želimo spremeniti. Klasičen način komunikacije je prikazan na sliki 5.1.

Rezultat klicanja zaposlenih je precejšnja izguba časa kontaktne osebe in posledično podjetja, poleg tega pa vzdrževalci lahko prepotujejo večjo razdaljo če so klicani na objekt, ki se nahaja izven njihovega razumljivega dosega.

5.2 Sprememba poslovnega procesa

Postopek spremembe je podrobno opisan v uvodu četrtega poglavja. Z uporabo naše rešitve optimiziramo komunikacijo med kontaktno osebo v pisarni in vzdrževalci na terenu.



Slika 5.2 Diagram prikazuje aktivnosti poslovnega procesa v primeru implementacije naše rešitve.

Komunikacija s klici preko mobilnega telefona se nadomesti z uporabo celostne rešitve, ki zaobjema spletno in mobilno aplikacijo, ter uporabnikom transparentno logiko v oblaku. Slika 5.2 prikazuje aktivnosti poslovnega procesa z implementacjo naše rešitve.

5.3 Predstavitev in uvedba rešitve v podjetju

Za uvedbo aplikacije v podjetje smo se dogovoril z vodjo servisne službe, preden smo začeli z razvojem. Za podjetje smo pripravili kratko predstavitev, ki se je odvila na ponedeljkovem jutranjem sestanku celotne servisne službe. Prikazali smo, kako aplikacija deluje, katere podatke zbira in kaj se v procesu testiranja rešitve pričakuje od vzdrževalcev in kontaktne osebe. Kratka navodila uporabe aplikacije smo naknadno poslali na e-poštni naslov vodje servisne službe z namenom, da so dostponi na zahtevo. Predlog rešitve smo začeli testirati 11.12.2017 in končali 5.1.2018, to je obsegalo 13 delovnih dni. Testiranje je potekalo ob delavnikih v dopoldanskem delovnem času, ki je praviloma od 7.00 do 15.00. Prve tri dni, do vključno 13.12.2017, smo namenili uvajanju sodelujočih v uporabo naše rešitve, preostali čas (10 dni), pa smo rešitev dejansko testirali in zbirali meritve. V obdobju uvajanja smo reševali probleme, povezane z uporabo aplikacije odgovarjali na vprašanja sodelujočih, ter se ukvarjali z nastavitvami mobilnih telefonov.

V testiranju celostne rešitve so sodelovali trije vzdrževalci in vodja servisne službe. Vsakemu od vzdrževalcev smo namestili mobilno aplikacijo. Pri tem smo imeli manjšo težavo. Vzdrževalci imajo službene mobilne telefone HTC Desire 820 ali 825. Omenjeni mobilni telefoni brez dodatnih aplikacij ne omogočajo brskanja po sistemskih datotekah znotraj naprave. Edini način brskanja po mobilni napravi je, da v nastavitvah vklopimo način za razvijalce in tudi takrat lahko sistemske datoteke samo beremo, izvršljivih datotek (*apk*) kot je npr. naša aplikacija, ni možno zagnati. Težavo smo rešiti tako, da smo vsakemu od vzdrževalcev inštalacijsko datoteko delili preko portala Google Drive. Ob prenosu datoteke na napravo, je datoteka dostopna v prenosih. Iz direktorija prenosi je izvršljivo datoteko možno zagnati in aplikacijo naložiti brez težav. Spletna aplikacija namenjena kontaktni osebi, je formata *html* in ne potrebuje namestitve, zato pri predaji spletne aplikacije nismo imel težav. Spletna aplikacija, kot že omenjeno, teče lokalno v podjetju.

5.4 Rezultati testiranja

V tej nalogi izboljšujemo čas, potreben, da informacija o alarmu na objektu pride do primernega vzdrževalca. Primeren vzdrževalec je tisi, ki je v razumnem dosegu do objekta z alarmom in v svoj delavnik lahko sprejme še eno obveznost. Da lahko objektivno pri-

merjamo učinkovitost servisne službe med obdobjem brez uporabe rešitve in obdobjem z uporabo rešitve, bi potrebovali čas, potreben za uspešno komunikacijo iz obeh obdobij. Rešitev smo zasnovali tako, da potrebne časovne žige zapisujemo v podatkovno bazo. Podjetje teh časov ne hrani, saj vsa komunikacija med kontaktno osebo in vzdrževalci poteka preko klicev na mobilni telefon. Čase izven obdobja uporabe rešitve smo poizkušali izmeriti sami s štoparico, vendar brez uspeha. Ob treh različnih priložnosti merjenja nismo dobili primernih rezultatov. Vodja servisne službe, ki večino časa opravlja vlogo kontaktne osebe in pri katerem smo poizkusili ročno izmeriti čas, je zelo zaposlen. Zaradi tega mu je bilo nemogoče s štoparico slediti toliko časa, da bi izmerili zadostno število alarmov za primerjavo. Prav to je bila dodatna motivacija za vpeljavo naše rešitve, ki kontaktno osebo razbremeni dodatnih opravil.

Zaradi neuspešnega merjenja časov izven obdobja uporabe rešitve smo pridobili druge podatke, s katerim lahko do neke mere objektivno primerjamo čase med uporabo rešitve. Iz pogovora z vodjo in pregleda dnevnika klicev službenega telefona, s katerega so opravljeni vsi klici, smo ugotovili, da največkrat opravi dva klica, preden informacija o alarmu doseže primernega vzdrževalca. To pomeni, da je bil zahtevek za delo zavrnjen enkrat. Nekajkrat je zahtevek za delo sprejet takoj, kar pomeni en opravljen klic, najmanjkrat pa se zgodi, da je zahtevek za delo zavrnjen dvakrat, kar pomeni, da vodja opravi tri klice. Dnevnik klicev smo pregledali za obdobje dveh delovnih dni in med podatki ni zabeleženo, da bi vodja kdaj opravil več kot tri klice, prav tako pa je to potrdil tudi sam. Vsak ponoven klic pomeni, da je klicani vzdrževalec zavrnil zahtevek za delo. Problem zavrnjenega zahtevka, je po besedah vodje predvsem v tem, da mora iskati telefonsko številko naslednjega vzdrževalca. Iskanje številke vzdrževalca terja svoj čas, poleg tega pa se dogaja, da med iskanjem vodjo v pisarni zmoti nekaj drugega in se obveščanje primernega vzdrževalca podaljša za precej časa. S pomočjo vodje smo izmed vseh klicev izbrali podatke tistih, za katere smo prepričani, da so bili opravljeni za namen obveščanja vzdrževalcev o alarmih. Časi pri alarmih z več kot enim klicem so prikazani približno, v minutah. Dnevnik klicev namreč ne omogoča prikaza časa med dvema klicema do sekunde natančno. Zbrani podatki so prikazani v tabeli 5.1.

Prikazani čas za posamezen alarm je seštevek časov klicev, opravljenih, da je informacijo o alarmu prejel primeren vzdrževalec. K seštevku, v kolikor je bil opravljen več kot en

klic, je prištet tudi čas med klici. Kljub majhnemu vzorcu opazimo, da se razlika v času drastično veča, v kolikor je opravljen več kot en klic. Vzorec osmih meritev izven časa uporabe naše rešitve je premajhen, da bi ga lahko uporabili za primejavo z meritvami znotraj časa uporabe rešitve, pa vendar nam da referenčne vrednosti, po katerih se lahko zgledujemo.

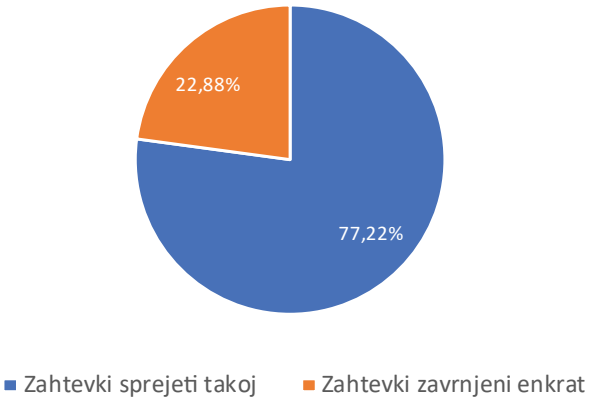
Alarm	**Število zavrnitev**	**Čas**
1	0	38 sek
2	0	49 sek
3	0	59 sek
4	1	2 min
5	1	3 min
6	1	3 min
7	1	5 min
8	1	8 min

Tabela 5.1 Časi, zbrani izven obdobja uporabe reštve

Vprašanji, ki sta vključeni v vsak pogovor, sta: *Kje se nahajaš?* in *Ali imaš čas opraviti še eno opravilo?*. Kontaktna oseba ob začetku delovnega dne pozna približno lokacijo vzdrževalcev, tekom dneva pa temu ni več tako. Kljub temu da časi iskanja vzdrževalca niso ekstremno dolgi, velja dejstvo, da izbira vzdrževalca preko telefonskega klica ni nujno najbolj optimalna, ker ne poznamo situacije ostalih vzdrževalcev.

S klikom v spletni aplikaciji se vsem zgoraj navedenim obremenitvam in težavam izognemo. Rešitev kot celota beleži štiri čase ob dogodkih, ko: je alarm sprožen preko spletne aplikacije s strani kontaktne osebe, je zahtevek za delo zavrnjen s strani vzdrževalca, je zahtevek za delo sprejet s strani vzdrževalca, je zahtevek za delo opravljen oziroma alarm izničen. En alarm lahko vsebuje tudi več časovnih žigov, v primeru da eden ali več vzdrževalcev zavrne zahtevek za delo. Zavrnitev zahtevka za delo v mobilni aplikaciji je ekvivalent ponovnemu klicu kontaktne osebe. V sklopu te naloge smo so osredotočili samo na čase od sprožitve alarma do sprejetja zahtevka. Čas, ki je potreben za odpravo alarma, ni pomemben, poleg tega pa je le ta odvisen od napake na objektu.

V sklopu testiranja naše rešitve je bil s spletno aplikacijo sprožen 101 alarm. Izmed vseh poslanih zahtevkov za delo jih je bilo 78 sprejetih takoj, 23 pa jih je bilo zavrnjenih enkrat. Med testiranjem zahtevek nikoli ni bil zavrjen več kot enkrat. Slika 5.3 prikazuje delež zahtevkov, ki so bili sprejeti takoj, in delež zahtevkov, ki so bili zavrnjeni enkrat.

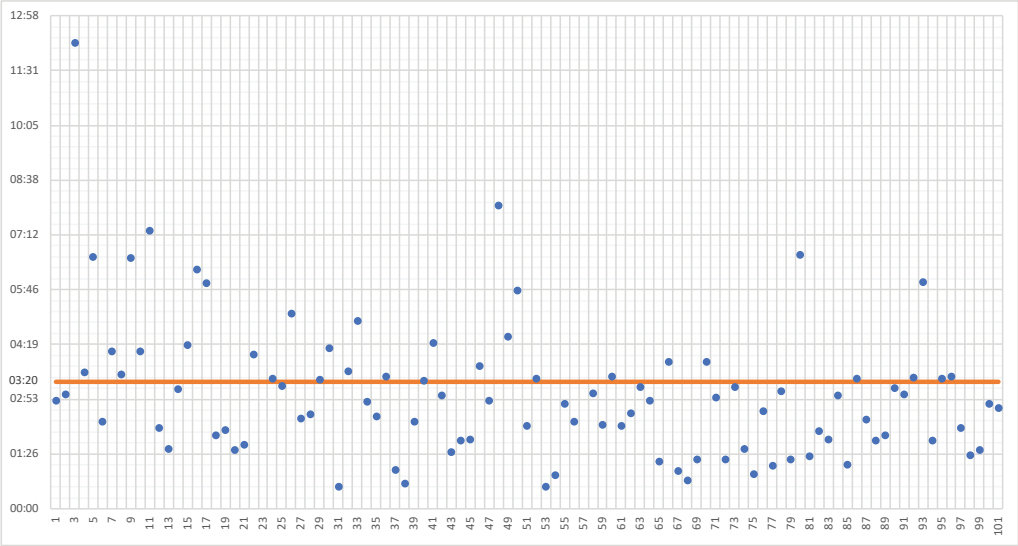


Slika 5.3 Deleži zahtevkov glede na število zavrnitev

Ta podatek nam da vedeti, da rešitev pravilno rešuje del zadanega problema. Ker je zahtevek za delo vedno najprej poslan najbližjemu, se zavrnitev zgodi samo v primeru, da je najbližji vzdrževalec v trenutku prejema zahtevka prezaposlen. Po pogovoru z vodjo servisne službe smo ugotovili, da je glavni razlog za zavrnitev zahtevka za delo neprimerna lokacija vzdrževalca glede na objekt z napako. Rezultat zavrnjenega zahtevka je opravljanje ponovnega klica s strani vodje. Po njegovih besedah ponovno klicanje opravlja v približno polovici primerov. Z uporabo aplikacije smo ta odstotek zmanjšali, poleg tega pa je zavrnjeni zahtevek vodji transparenten, saj rešitev sama najde naslednjega vzdrževalca.

V povprečju je bilo v času uporabe naše rešitve sproženih deset alarmov dnevno. V začetku testiranja se na podlagi odzivnih časov opazi, da vzdrževalci še niso popolnoma sprejeli novega načina obveščanja. Časi so daljši in opazimo lahko večja odstopanja od povprečja. Graf na sliki 5.4 prikazuje čase alarmov, ki so bili zajeti tekom testiranja. Osredotočamo se na čase od poslanega zahtevka za delo s strani kontaktne osebe do sprejetja zahtevka s strani vzdrževalca. Časi so prikazani kronološko z leve proti desni. Vertikalna os nam prikazuje čas trajanja v načinu *mm:ss*, horizontalna pa zaporedno

številko alarma.



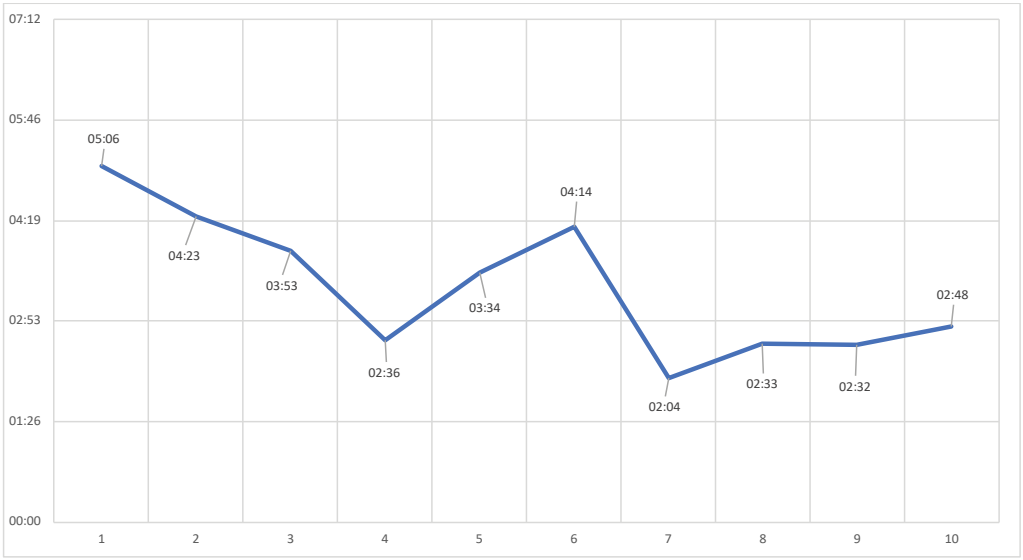
Slika 5.4 Časi uspešne komunikacije med kontaktno osebo in vzdrževalci

Povprečen čas vseh uspešno posredovanih zahtevkov v obdobju testiranja, ki ga prikazuje črta oranžne barve, je 03.20*(mm:ss)*. Najdaljši čas, ki smo ga zabeležili, pripada alarmu z zaporedno številko 57 in je enak 23:33. Zaradi boljše berljivosti podatka o najdaljšem zabeleženem času v grafih ne prikazujemo. Daljši časi, ki silijo krepko izven povprečja, so po besedah sodelujočih rezultat spleta različnih okoliščin, ki privedejo do tega, da vzdrževalec svojega mobilnega telefona ne pregleduje frekventno. Tipičen primer je v avtu pozabljen mobilni telefon. Najkrajši izmerjeni čas je 00:34.

V drugi polovici testiranja so časi krajši in veliko manj razpršeni, kar nakazuje na dejstvo, da se uporabniki privajajo k uporabi rešitve. Graf na sliki 5.5 prikazuje povprečne čase po dnevih. Najdaljši čas je bil izmerjen šesti dan, kar se opazi pri poteku premice. Prav tako je bil šesti dan izmerjen najkrajši čas. Kljub krepko izstopajočim vrednostim izven povprečja povprečni čas odziva vztrajno pada in se proti koncu testiranja ustali med 02:00 in 03.00.

Za testiranje smo imeli na razpolago določeno število dni. Lahko rečemo, da bi se čas odziva z veliko verjetnostjo še zmanjšal, v kolikor bi rešitev v podjejtu uporabljali dlje. Največjo razliko v času celotnega poslovnega procesa je opazil vodja, ki je med vsemi

udeleženci najbolj zaposlen in nosi največjo odgovornost. Vodja ima s proženjem alarma preko spletne aplikacije enako dela, ne glede na to, koliko časa informacija dejnsko potrebuje, da doseže primernega vzdrževalca, in koliko zavrnitev se medtem zgodi.



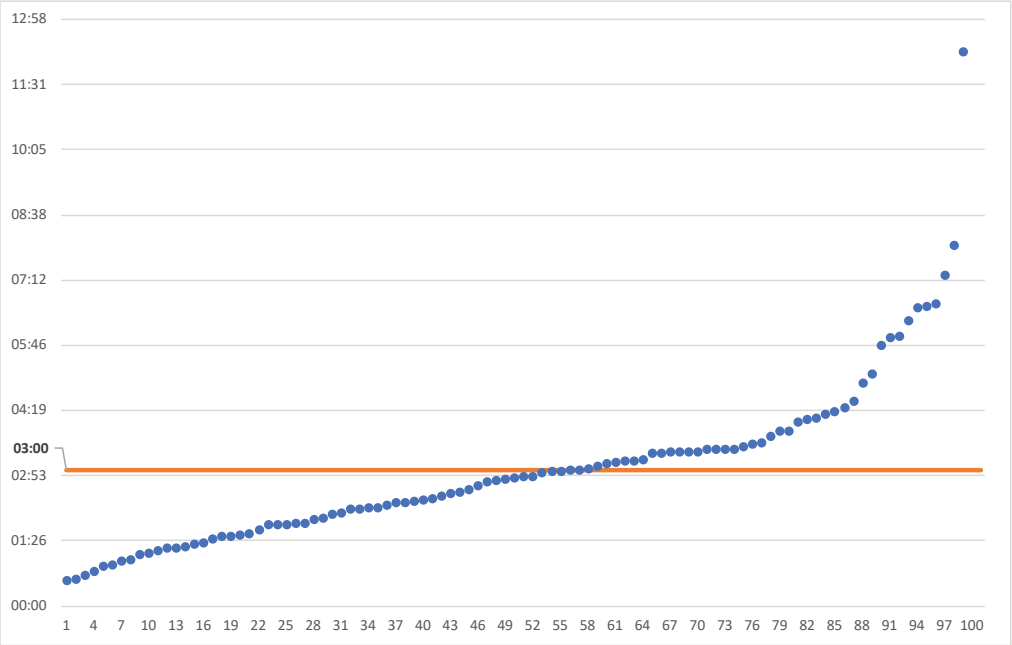
Slika 5.5 Povprečni časi odzivov po dnevih

Podatke, ki smo jih pred začetkom testiranja pridobili iz dnevnika službenega mobilnega telefona kontaktne osebe, smo uporabili kot referenčne vrednosti za prikaz približne količine prihranjenega časa z uporabo naše rešitve. Graf na sliki 5.6 prikazuje podatke iz grafa na sliki 5.4 le da so urejeni po velikosti. Horizontalna oranžna črta prikazuje povprečje vrednosti iz tabele 5.1, ki znaša 175,75 sekund, vendar smo ga zaradi približnih podatkov zaokrožili na 180 sekund (3min). Pod povprečnim časom je bilo v času testiranja izmerjenih 57 časov, kar je več kot polovica vseh izmerjenih časov.

V testnem podjetju sistem SCADA letno sproži okoli 51.000 alarmov. Po pogovoru z zaposlenimi smo izvedeli, da niso vsi alarmi namenjeni servisni službi. Med alarme spadajo: obvestila o stanju naprav, ki jih sistem SCADA generira z določeno frekvenco, testna sporočila, poročila o porabi, popisi števcev ter dejanske napake na napravah. Bolj podrobnega vpogleda v številke alarmov v posameznih kategorijah nismo dobili.

Po številkah, ki smo jih pridobili med testiranjem in po pogovoru z vodjo servisne službe, lahko rečemo, da testno podjetje na letni ravni prejme in razreši približno 4000 alarmov.

Za vsakega od teh alarmov je potrebno poskrbeti, da informacija o okvari na objektu pride do primernega vzdrževalca. Zaradi pomanjkljivosti podatkov med uporabo klasičnega obveščanja ne moremo izračunati natančno, koliko bi z uporabo naše rešitve prihranili na času. Lahko pa rečemo, da smo razbremenili kontaktno osebo in da je naša rešitev v približno polovici primerov hitrejša od prej uporabljene metode ter je primerna za implementacijo v dejansko podjetje.



Slika 5.6 Prikaz časov, urejenih po velikosti v primerjavi s povprečnim časom izven uporabe naše rešitve

Po koncu obdobja testiranja je vodja servisne službe skupaj z vzdrževalci podal mnenje o uporabi celostne rešitve. V ta namen smo pripravili kratek vprašalnik, s pomočjo katerega smo zbrali subjektivne ocene različnih lastnosti delovanja rešitve. Uporabniki so ocenili posamezne lastnosti z oceno od 1 do 10, s tem da 1 pomeni zelo slabo in 10 odlično. Lastnosti in ocene posameznega uporabnika so predstavljene v preglednici 5.2.

Poleg vprašalnika smo se z udeleženci testiranja pogovorili osebno. Vprašanja, ki smo jih zastavili, so se nanašala predvsem na željene spremembe v delovanju aplikacije, na izgled uporabniškega vmesnika in na smiselnost uporabe takšne rešitve. Vzdrževalci so mobilno aplikacijo pohvalili z vidika hitrega dostopa do potrebnih informacij, enostavno-

sti uporabe in razumljivosti. Negativne opazke so bile predvsem v smeri porabe baterije in občasno počasnega delovanja.

Lastnost	**Vodja**	**Vzdrževalec A**	**Vzdrževalec B**	**Vzdrževalec C**
Enostavnost uporabe	10	9	8	9
Razumljivost	10	10	8	8
Uporabnost	9	9	9	9
Učinkovito delovanje	8	8	8	8
Splošen vtis	9	8	8	9
Občutek prihranjenega časa	9	8	9	9

Tabela 5.2 Prikaz ocenjenih lastnosti uporabe s strani posameznega uporabnika

Že med razvojem in tudi med dejanskim testiranjem je bil vodja servisne službe pozitivno presenečen nad funkcionalnostmi, ki jih pridobi z uporabo rešitve. Komentarje o spletni aplikaciji je podal vodja, saj je on največkrat zasedal vlogo kontaktne osebe. Po njegovih besedah je največja prednost uporabe prav ta, da lahko ažurno spremlja lokacijo vzdrževalcev in njihove obremenitve v delovnem dnevu. Naloga vodje servisne službe ni samo razporejanje opravil med vzdrževalce, pač pa skrbi še za druge stvari, ki se tičejo upravljanja z objekti. Iz tega pa izhaja naslednja prednost in hkrati tudi pomanjkljivost aplikacije. Prednost v uporabi spletne aplikacije je ta, da čas, porabljen s strani vodje za iskanje primernega vzdrževalca, zmanjšamo na le nekaj sekund in se tako vodja lahko osredotoči na druge stvari. Poleg tega je vodja servisne službe več časa dosegljiv na mobilni telefon in tako lahko sprejema klice strank.

Kljub temu da bi praviloma v podjetju moral biti konstantno nekdo prisoten v vlogi kontaktne osebe, se lahko zgodi, da zaradi različnih obveznosti temu ni tako. Spletna aplikacija, s pomočjo katere sprožimo alarm, je bila zasnovana izključno za uporabo na računalniku, poleg tega teče lokalno v podjetju. Alarma tako ni možno sprožiti brez računalnika, na katerem teče aplikacija, kar je v uvajalnem obdobju predstavljalo težave. Naslednja pomanjkljivost, ki jo je izpostavil vodja, je ta, da z vzdrževalci preko aplikacije ne more komunicirati drugače kot s sproženim alarmom. Želel bi, da sporočilo v zahtevku za delo lahko oblikuje sam. Razlog za to je, da informacija o nepravilnem delovanju objekta v podjetje lahko pride tudi preko e-pošte ali klica stranke. Z možnostjo

oblikovanja sporočila v zahtevku za delo bi kontaktna oseba lahko dodala bolj podroben opis. Prejemnik bi na ta način bolj natančno vedel, v čem je problem na objektu.

6 Sklepne ugotovitve

V času razvoja mobilne aplikacije smo bili konstantno v kontaktu z vodjo servisne službe in zelo malo z vzdrževalci, ki so med testiranjem uporabljali mobilno aplikacijo. Po našem mnenju je bila to napaka, saj so bili na dan pričetka testiranja vzdrževalci presenečeni. Potrebovali so dalj časa, da so uporabo aplikacije dejansko sprejeli in jo vestno začeli uporabljati. Za učinkovito uporabo mobilne aplikacije mora vzdrževalec svoj mobilni telefon frekventno pregledovati, da opazi nove zahtevke za delo. V začetku je bila to težava, ki je postopoma izginjala. Logična rešitev za ta problem je, da v aplikacijo implementiramo zvočni signal. Naprava mora imeti za delovanje aplikacije dostop do internetne povezave in GPS sprejemnika, kar pa porabo baterije, kljub temu da aplikacija teče v ozadju, precej poveča. Obremenjevanje naprave z aplikacijo se v nekaterih primerih pozna tudi pri uporabi mobilnega telefona za druga opravila. Rešitev za probleme v zvezi trajanja baterije in s počasnega delovanja se skriva v optimizaciji celotne kode, virov in v logiki shranjevanja podatkov.

Med testnim obdobjem smo sami opazili še nekaj možnih izboljšav. Aplikacija teme-

lji na brezplačni oblačni storitvi Google Firebase. Omenjena brezplačna storitev teži k zakasnitvi delovanja v primeru velike obremenitve v kratkem času. To pomeni, da naše zahteve ne bodo trenutno obdelane, kar pa nam ni predstavljalo občutnejših težav. V kolikor bi želeli oblačno storitev brez zakasnitve, je ta povezana z dodatnimi stroški. Kot je že znano, mobilna aplikacija potrebuje za učinkovito delovanje internetno povezavo. Največkrat je v ta namen služil mobilni internet, nekajkrat pa tudi brezžična povezava na različnih lokacijah. Ko so vzdrževalci na poti, lahko naletijo na mesta, kjer je pokritje z mobilnim internetom slabo. Če je ravno v času poslanega zahtevka za delo eden izmed naslovnikov v vrsti brez možnosti internetne povezave, se na poslani zahtevek ne more odzvati in tako podaljša čas iskanja primernega vzdrževalca. Rešitev za ta problem je funkcija z implementacijo časovnika v oblaku, ki v primeru brez odziva po določeni pretečeni časovni enoti samodejno pošlje zahtevek naslednjemu v čakalni vrsti. Večjih težav zaradi omenjenega problema ni bilo.

Z razvojem in uporabo celovite rešitve smo pokazali, da je raba sodobnih tehnologij v poslovnih procesih primerna. Po mnenju uporabnikov smo problem reševali praktično, učinkovito in na razumljiv način. Prav tako lahko rečemo, da je produkt z upoštevanjem izboljšav primeren za produkcijo. Naslednji logičen korak v razvoju je integracija s sistemom SCADA. Na ta način bi postopek obveščanja popolnoma avtomatizirali. Potrebno se je zavedati, da ta rešitev rešuje problem na kar se da splošen način. Če bi s prototipom želeli nadaljevati v produkcijo in predstavljati konkurenco na trgu, bi se morali osredotočiti predvsem na delovanje potencialnega podjetja, v katerem bi rešitev uporabljali. Tako bi splošen prototip dopolnili s specifičnimi zahtevami in problem v izbranem podjetju naslovili bolj učinkovito.

Literatura

[1] Statistika o uporabi pametnih mobilnih telefonov, Dosegljivo: http://mobilemarketingmagazine.com/24bn-smartphone-users-in-2017-says-emarketer, [Dostopano: 27. 12. 2017].

[2] Uporaba mobilne tehnologije v poslovnem svetu, Dosegljivo: https://www.apperian.com/mam-blog/mobile-apps-can-transform-business-processes/, [Dostopano: 16. 1. 2018].

[3] Primer uporabe sistema scada v proizvodnji, Dosegljivo: https://en.wikipedia.org/wiki/SCADA, [Dostopano: 15. 12. 2017].

[4] Prednosti in slabosti sistema scada, Dosegljivo: http://lrf.fe.uni-lj.si/e_rio/seminarji1415/Seminar17.pdf, [Dostopano: 12. 12. 2017].

[5] Prednosti in slabosti sistema scada, Dosegljivo: https://www.fvv.um.si/DV2012/zbornik/varnostno_obvescevalna_dejavnost/strmsek.pdf, [Dostopano: 13 12. 2017].

[6] Sistemi scada, Dosegljivo: http://www.scadasystems.net/, [Dostopano: 6. 12. 2017].

[7] Obvladovanje alarmov, Dosegljivo: https://en.wikipedia.org/wiki/SCADA#Alarm_handling, [Dostopano: 15. 12. 2017].

[8] Problemi v procesu alarmiranja, Dosegljivo: http://www.recursionsw.com/2012-10-17_SCADA_System_Alarm_Notification_Options_for_Workforce_in_a_Plant_Environment.pdf, [Dostopano: 15. 12. 2017].

[9] Tasker, Dosegljivo: https://www.taskertools.com/eu/en/, [Dostopano: 9. 12. 2017].

[10] Mobilplan, Dosegljivo: https://mobilplan.nu/, [Dostopano: 9. 12. 2017].

[11] Recursion, Dosegljivo: http://www.recursionsw.com/, [Dostopano: 15. 12. 2017].

[12] Haversinova formula za izračun razdalje med dvema paroma koordinat, Dosegljivo: https://opcfoundation.org/developer-tools/specifications-classic/alarms-and-events/, [Dostopano: 23. 1. 2018].

[13] Platforma google firebase, Dosegljivo: https://firebase.google.com/, [Dostopano: 17. 12. 2017].

[14] Format za hranjenje podatkov json, Dosegljivo: https://www.json.org/, [Dostopano: 17. 12. 2017].

[15] Programski jezik javascrpit, Dosegljivo: https://developer.mozilla.org/en-US/docs/Learn/JavaScript/First_steps/What_is_JavaScript, [Dostopano: 17. 12. 2017].

[16] Haversinova formula za izračun razdalje med dvema paroma koordinat, Dosegljivo: http://www.movable-type.co.uk/scripts/latlong.html, [Dostopano: 23. 12. 2017].

[17] Označevalni jezik html, Dosegljivo: https://www.w3schools.com/html/html_intro.asp, [Dostopano: 17. 12. 2017].

[18] Programski jezik javascrpit, Dosegljivo: https://sl.wikipedia.org/wiki/IntelliJ_IDEA, [Dostopano: 17. 12. 2017].

[19] Android operacijski sistem, Dosegljivo: https://en.wikipedia.org/wiki/Android_(operating_system), [Dostopano: 17. 12. 2017].

[20] Android operacijski sistem, Dosegljivo: https://developer.android.com/studio/index.html, [Dostopano: 17. 12. 2017].

[21] Način shranjevanja podatkov shared preferences, Dosegljivo: http://codetheory.in/android-application-data-storage-sharedpreferences/, [Dostopano: 23. 12. 2017].

[22] Odprtokodna knjižnica gson, Dosegljivo: https://github.com/google/gson, [Dostopano: 23. 12. 2017].

[23] Zahtevek post, Dosegljivo: https://www.w3schools.com/tags/ref_httpmethods.asp, [Dostopano: 23. 12. 2017].

[24] Petrol d.d. - o podjetju, Dosegljivo: http://www.petrol.si/o-podjetju/petrol/o-petrolu, [Dostopano: 5. 12. 2017].

[25] Petrol d.d. organizacijska enota energetske rešive - o podjetju, Dosegljivo: http://www.eltec-petrol.si/domov/, [Dostopano: 5. 12. 2017].

[26] Plc krmilnik uporabljen v testnem podjetju, Dosegljivo: http://www.eltec-petrol.si/daljinska-energetika/tehnologija/sistemi-regulacije-procesov-eltec-tp-06-eltec-tp-07-eltec-tp-08-in-eltec-tp-09-plc/, [Dostopano: 27. 12. 2017].

[27] D. D. Damayanti, H. Rachmat, D. S. Atmaja, Design of integrated scheduling and automated controlling for surface treatment process using supervisory control and data acquisition (scada), in: 2013 IEEE International Conference on Industrial Engineering and Engineering Management, 2013, pp. 1577–1581.

[28] A. DANEELS, What is scada, International Conference on Accelerator and Large Experimental Physics Control Systems, Trieste, Italy (1999) 339–343.

[29] A. Gligor, T. Turc, Development of a service oriented scada system, Procedia Economics and Finance 3 (2012) 256–261.

[30] D. Hadžiosmanović;, D. Bolzoni, P. H. Hartel, A log mining approach for process monitoring in scada, Int. J. Inf. Secur. 11 (4) (2012) 231–251.
url: http://dx.doi.org/10.1007/s10207-012-0163-8

[31] A. A. Sallam, O. P. Malik, Electric distribution systems, Vol. 68, John Wiley & Sons, 2011.